A Standard Methodology for the Interoperability of Heterogeneous Information Sources

By

Jehad Saleh Ashir

A Dissertation Presented to the
Faculty of Computing Sciences and Engineering

De Montfort University

In Partial Fulfillment of the
Requirements for the Degree

DOCTOR OF PHILOSOPHY
Institute of Mathematical and Simulation Sciences
Faculty of Computing Sciences and Engineering

September 2001

Abstract

A temporal layer to facilitate the interoperability between the heterogeneous information sources is presented. The way in which this layer can provide better connectivity, security, manageability and robustness to the different disparate information sources in the Internet is studied. In addition, a definition of the various web enabling parameters required to undertake the process of information sources interoperability is given.

A standard is developed (based on a technical perspective) of the system architecture requirements to interoperate between heterogeneous databases in the light of the current Internet advances. Such approaches attempt to facilitate the integration of distributed heterogeneous databases in a dynamic incremental manner. It provides the shape, feel and the look of a universal database viewing architecture. A study is undertaken of the different issues relating to metadata hosting that forms the basis for integrating and aggregating distributed heterogeneous information sources. Further, the design of a linkage between distributed heterogeneous information sources and permitted distributed user groups is presented. The number of indispensable services such as the universal syntactic/semantic data dictionary that categorizes the meanings and formats of different components of the cooperating information sources discussed.

The major contribution of the work in this dissertation is a study of the requirements for an interoperability for Database Management Systems that are able to handle unification for the various dissimilar processes between the heterogeneous distributed information sources. The work requires the development of a solid strategy to be followed by different DBMS vendors on the data exchange formats in a unified manner. There are also many processes that require standardization and unification such as data types, code pages, indexing techniques, and application to application calling techniques.

As a subsequent contribution to the main research, a unified methodology is created to advertise any schema type that is accessible as an information space using the web browser as a local communicator between global information producers and consumers. The proposed solution requires a specialized proxy server responsible for communications between cooperating information producers and a specialized search engine responsible for keeping the different information consumers informed about the available information space.

The goal of this research is the design and partial implementation of adaptive methodologies and toolkits for the interoperation between heterogeneous information sources in large-scale and rapidly growing network environments. The solution is to implement a prototype responsible for the delivery of information for the normal information consumers who do not really care about what schema type they are accessing or do not have knowledge and/or experience of multiple schema usage. The most important aspect is to pick the information they need from disbarred information sources, merge them in their schema, manipulate them and obtain the reports they need. The most comprehensible shape for the information is when it is a basic non-normalized shape. These features allow the information consumer to insert the data in any local schemas regardless of their type.

The prototype aims at developing toolkits which make use of the treelike nature of the different database schemas and their different schema components to facilitate the interconnection between information consumers and information producers. The main purpose of utilizing the treelike structure of the different schemas is to make the concept of interoperability as simple as possible. The bond which forms the relationship between different schemas is a relatively straightforward task. Necessary database services related to the reliability, maintainability and ease-of-use of the heterogeneous database interoperability process are studied. The prototype consists of several interrelated components that can join information from disparate information repositories in a transparent manner and reflect the gathered information to the information consumer in a shape that is clear and transparent.

The main services the prototype provides includes: (i) the registration of shareable information source portions; (ii) the propagation of shareable information sources to global information consumers; (iii) metadata management; (iv) local schema constraint enforcement; (v) heterogeneous database schema migration; (vi) data replication management and unified database security management. This thesis presents the design of the proposed prototype and its partial implementation. The prototype is responsible for forming the interoperation layer that simplifies the heterogeneous information sources integration.

The approach proposed, provides a mechanism that enables database application users to be informed about the available information space they can gain from global information producers. It also enables them to share information with other information holders in a transparent, expandable and autonomous manner. The design of the prototype is accomplished in a way that encourages collaboration with metadata capable of web proxy servers so that they become heterogeneous databases accessible by web browsers.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Professor Jonathan Blackledge and Dr Mohammad Jaffar without whom I would never have completed this thesis.

Also, I wish to thank Yusuf Abdullah for the helpful and generous discussion during the implementation for some of the ideas for this research thesis.

I am also very grateful to my friends and colleagues from the Central Statistics Organization of the State of Bahrain for being a 'surrogate family' during the many years I stayed there and for their continued moral support thereafter.

I would also like to thank the people who offered essential moral and emotional support throughout this long project, in particular, to my family: mother Lulwa, brother Salah, sisters Aneesa, Dhabia, Fatema and my children Mohammed and Sara.

Finally, my wife Faiza assisted me in many ways; without her patients and encouragement, this work would never have reached completion. Hopefully, now I can spend less time in far-off places and more time with the people I most love.

Contents

Chapter 1

Introduction

The rapidly growing amount of information available over the worldwide networks and the distributed heterogeneous nature of the information is having a major impact on the field of information sharing. It is always true that any information certainly exists somewhere in the universe but reaching this information is not always a straightforward task. For this reason, knowledge distribution and discovery for the distributed heterogeneous databases are very important facilities that should exist in all cooperating databases to facilitate the data sources interoperation. As the information highway is rapidly expanded this requirement will be increasingly true. The proper management for the required information about the cooperating data sources is the major solution to the problem. Hence, the gathered information should be capable of giving global users the necessary knowledge about the provided information space that the information producer will offer.

Multidatabase researches are divided into the classic approaches, the federated approach and the distributed object management approach. The classic approaches for multi-database only depend on creating a global schema by which a common query language is normally used for updating the global repository. The federated approach relies on multiple import schemas and the customized integration at various multi-database levels is enforced by the local system. The heterogeneity problems are solved at the schema integration stage. This approach cannot scale well when new sources are to be added because the component schema cannot evolve without the consent from the integrated schema. The object management approach on the other hand has generalized the federated approach by modeling the heterogeneous databases as objects in an object space. This technique requires the definition of a common object model and a common object query language.

It is well known that databases are collections of related data stored electronically in storage media. Database management systems, which are a collection of programs, enable users to create and maintain databases. They are considered as general-purpose software systems that facilitate the processes of defining, constructing and manipulating information sources for various applications. Throughout their milestone database schemas, or as they are sometime called data models, come in a number of shapes in term of their storage technique. The first generation is considered to be the hierarchical schema data model. The second generation is network data models. The relational and the object-oriented data models are known as the third generation databases. So far the proposed data models fall into three different groups: object-based logical models, record based logical models, and physical models [Silberschatz97]. The three groups will be thoroughly discussed in the next chapter.

Originally, all databases where largely centralized in the sense that all the system components where resident at a single computer or site. The components include the database, the database management software and the secondary storage where all the operations are done. However, in recent years there has been a rapid move towards the distribution of computer systems over multiple sites that are interconnected via a communication network. This demand has led to the birth of the distributed database management systems DDBMSs.

Furthermore, the distributed databases are either homogeneous from similar types or heterogeneous from different types. In case they are homogeneous, there will be no need for

the homogenization process, which is only used when different types of databases are to be used together. In many situations, sites with different DBMSs (Relational, Hierarchical, Network, and Object-Oriented) exist and accessed by various numbers of local and global users. In addition, there is a great demand nowadays for distributing the databases because of the new changing business requirements. Building an interoperation mechanism between the different databases require knowledge about the interoperating data sources (i.e. database name, place, type, provided information space, etc.) and middle engine support. The main task of the middle engine is to act as a custodian between the cooperating distributed heterogeneous data sources so that the necessary parameters are exchanged easily between the cooperating databases. Also, the middle engine metadata knowledge will facilitate the work currently required by many management components. Among the components are: schema translation management, programming languages translation management, semantic inconsistency management, and other aspects related to the operating system and the communication protocols layers which are considered as outside the database realm research. In a large network of interoperable autonomous heterogeneous databases, the architecture should be flexible enough to allow negotiation to take place to establish grouping of databases. In such a large environment, it is important that databases be made aware of the other participate databases in an incremental and dynamic fashion [Bouguettaya94, Raschid94]. Users need to be incrementally and dynamically informed about the available information and where it is located.

In addition, each of the databases has their own constraints over the stored schemas and their attributes. Constraints in data modeling are divided into five different areas, which are: 1) domain constraints, 2) key and relationship constraints, 3) general semantic integrity constrains, 4) inheritance, implicit, and explicit constraints, and 5) state versus transition constraints [Elmasri94]. Constraints are considered as powerful procedures enforced by the DBMS on the substituted values entered by the users. Applying constraints in the distributed heterogeneous databases is a must by which retaining them as they appears at their place is forming a major factor of the success of the data interoperation. The cooperation of the distributed heterogeneous databases is assumed not to violate any of the five constraint types during the cooperating process. The cooperation process can be defined as a linkage between two non-related data sources for the sake of getting collaborated information. Basically, cooperation between information sources could be a relationship operation with other data sources, a unification process or an intersection process with the other data sources.

The overall objective of this dissertation is to present a new, modular and dynamic facility for the knowledge distribution and cooperation in the distributed heterogeneous databases to facilitate the interoperation mechanism between the data sources utilizing the current Internet advances. The distributed data sources should know about the other data sources willing to cooperate in sharing and exchanging information. The size of information domain provided by others is an important factor that should be known by which it would simplify the cooperation that can happen between the information sources. In order to know the behavior of the different database schemas a close study to the heterogeneous schemas was involved to get a close idea about the necessary parameters that should be known by the interoperation engine. The proposed engine also built to retain the distributed data sources own constrains which is a major issue to be included in the proposed interoperation engine design. Accordingly this research will design, partially implement, and test an experimental prototype within the framework of the interoperation engine experimental system.

As subsequent complementary objectives of this research is to design a unified security manager and other assistance services for the heterogeneous databases based on the deliverables of the main overall objective. Additionally, this thesis will study, develop, and standardize specifications for the technology necessary to support heterogeneous distributed database interoperability.

This dissertation is aimed at characterizing the necessary knowledge from the heterogeneous databases (i.e. relational, hierarchical, network, and object-oriented) and created the necessary interfaces and connection protocols that will cooperate in keeping users incrementally and dynamically informed about the available information space and where it is

located. This is because the static approach to user education cannot be considered as reasonable in an information environment that is rapidly growing. The experiment of this work is based on the relational data model schema, the IMS system schema, the IDMS system schema, and the O2 system schema respectively. In this dissertation the problem various dimensions has been researched and a prototype has been proposed, which forms the foundation of an important task that should be undertaken soon. The proposed prototype is aimd at simplifying the interoperation process of the heterogeneous databases by having a single point of access to different schema types. This process will simplify accessing the different schema types from outside. The proposed engine will unify the reflected information to the outside requestors regardless of the schema type.

1.1 Perspective in Next Generation Database Technology

The tremendous explosion of networking technology has led to the ubiquitous Internet and the large-scale accessibility of diverse information sources. Past studies in the mid 90s estimate that there are ten million hosts and twenty million users on the Internet with numbers doubling roughly every year and probably these days the number of users doubling each month. The interoperation of the numerous incompatible Internet protocols has also led to large number of users accessing the Internet and seeking information. This has resulted in a wide variety of hosts providing information of many types (statistical data, flat files, multimedia files, databases, etc.). Currently existing Internet browsers are mainly designed to support extracting HTML format files from all over the world. They do not distinguish between the information source types whether they are flat files or databases. Internet service tools have not adequately resolved these issues. The result is that the information consumers are left with the daunting tasks of resolving information heterogeneity and learning different information access methods. The development of a database-capable World Wide Web to link data sources is becoming a must these days. This will resolve many issues of which the data consumer may spend 75% of his time trying to resolve such as transferring the information of interest to his data pool. The achievement of such task has many prerequisites by which one is to make the heterogeneous data sources able to define themselves on the web so that others know them, which is the current objective of the work at the moment.

In an environment such as the Internet it is difficult for users to manage getting information of interest they are looking for in the light of the expansion in the data sources. Users in this case need to know (*what*) available data is of interest. It will be helpful to them if they know where the information is, if this will effect their application, which the prototype conducts the (*where)* in a transparent manner. The (w*hat)* part is been addressed by the FINDIT [Bouguettaya91], which the prototype considers as an extension. Also, the problem has been looked to from a different elevation in a way to make the prototype behave as routing device for the cooperating data sources. In general terms, the problem is typically associated with the size of the available data, the heterogeneous nature of the data, and the extraction and handling of the necessary information that will form the knowledge which will facilitate the interoperability of the distributed heterogeneous databases.

In addition, the existing Internet client browsers are still considered as general purpose extracting applications. They mainly deal with server browsers to extract the required information from a database stored in the server known as *proxies* provided by Internet service providers. Although browsers succeeded in dealing with number of operating systems, they are still not dealing with the interoperation of the heterogeneous databases. At the present time, there is a great demand for enabling the existing browsers to support the heterogeneous databases. Systems such as FINDIT [Bouguettaya95a, Bouguettaya95b] and Sybil [King] have addressed the problem of the knowledge discovery in large-scale heterogeneous databases. But, still dealing with the heterogeneous databases needs to be done in a systematic manner. By so the different constraints supported by the different cooperating database schemas are retained, and no doubt the near future browsers will take care of the heterogeneous databases.

Current Internet searching engines totally handle non-structured information sources based on indexing and keyword searching. There are nowadays numerous search engines that can be found on the web handling the unstructured information sources employing different

techniques to build indices some of which are more intelligent than others. Users usually spend horrendous amounts of time sifting through a huge amount of pages. In contrast with this the proposed prototype has been mainly designed to managed fully structured metadata related to distributed life running database systems having schema type (i.e. relational, hierarchical, network or object-oriented). The result of a searching operation is displayed in a unified manner as the result of searching the web in the normal unstructured information sources. The prototype is designed to act as another level of integration between the information producers and information consumers in a seamless way using a specialized heterogeneous schemas capable proxy server and search engine.

A key aspect of making data being useable across heterogeneous databases involves schema or partial schema unification, and it is imperative to develop a method that can perform this task with a dynamic way, useable to both the normal and novice users. In this dissertation the term "unification" or "cooperation" to refer to the interoperation mechanism between the heterogeneous data sources has been used. Metadata means objects that represent the structural part of the database or the conceptual schema itself. Metadata is descriptive information about the structure and meaning of data and applications and processes that manipulate data. Rather than unifying complete schema with each other, which is a difficult and costly process, partial schema unification is considered in the proposed prototype selectable by demand from the provided information spaces. One of the fundamental problems to be solved before one can successfully unify between schemas or partial schemas is that of metadata handling between the interoperating information sources. In the database context, this is referred as data about data.

1.2 Scope of the Thesis

This thesis presents the design and discusses the partial implementation of a prototype responsible to facilitate the interoperation process between different distributed heterogeneous database schemas, namely the Interoperation Engine, which is abbreviated as IE. It is consisting of two stages: the first stage is the metadata knowledge base and the second stage is the supporting engines to derive the interoperation. This prototype illustrates the ability to automate the interoperability of the distributed heterogeneous databases in a transparent, cooperative and incremental. The aim behind such a design is to make the information consumers get more accurate information from multiple information producers, as well as, create an atmosphere by which the information producers can easily advertise about the information pool they would like to make it known to the public. Additionally, the prototype is considering the full autonomy to be on the hand of the information owner over their information pools. The prototype also includes the description of the algorithms, which makes the interoperation process between the different schema types possible. The prototype is mainly dealing with several components that gather information from structured data sources, and present the gathered information in terms of the consumer's local DBMS and interface description. The main component services that the IE prototype addresses include the information producers' information source registration with all the related database components, the related permissions on the information source, and the users profiles the access should be given to them. This thesis will present the initial prototype effort by a work-through example with the objective of demonstrating the database interoperability process and highlighting the different requirements by investing the existing interoperation facilities.

The overall work in the research of this area is directed towards achieving a number of strategies that are necessary for accomplishing the interoperability process among the distributed heterogeneous databases. The initial main strategies that are undertaken during the work in this field are:

1. The definition of the metadata related to the heterogeneous schemas that will contribute in the interoperation and dynamically allow new information sources to be incorporated into the cooperation.
2. Build the strategy of metadata registration, which acts as the infrastructure that will be used by the information producers to advertise the existence of certain information.

3. Binding the local schemas with the global cooperating schemas so that incremental addition in the global cooperating schemas should not affect the ongoing tasks.
4. Provide the authorized information consumer with the ability to bind their own information sources with the global information space that has the necessary authority.
5. Provide the necessary management services for the interoperation such as security services; syntactic/semantic data dictionary services; history tracking and replication services.
6. The requirements towards providing database services for current Internet browsers.

Throughout this thesis, a number of existing approaches are used to make sure that the prototype has been tested and considered to be an extension for them, does not have any major conflicts or repetitions with regard to what has already been achieved in the area.

Also, throughout the design phase of the IE proposed prototype it is intended to initiate a number of mechanisms and standards to be followed by the information owners to interoperate with other information sources having similarities. The standard part of the IE defines the following:

1. A common way for the database information producers and consumers to access data and perform data sharing operations in a unified manner.
2. A mechanism for keeping the information consumers informed about any changes in the global information space they might already get access on.
3. A standard "metadata and data" handling mechanism and a set of database interoperation standards.
4. A standard and mechanism to be followed by the database producers to follow when exchanging information with other non-similar information sources.

1.3 The main contribution of the thesis

In general, the overall contribution of the work in the area of database interoperability is to setup the requirements towards having interoperability capable Database Management Systems that are able to handle the interoperability requirements for the various dissimilar processes between the heterogeneous distributed information sources. A summary of the original contributions undertaken in this thesis is as follows:

1. *Research systems in the area of information interoperability and the different schema capabilities and limitations to interoperate with each other.*
2. *A framework responsible for the heterogeneous information sources interoperability in a dynamic manner using the Internet facilities.*
3. *An architecture of a prototype designed from scratch to provide maximum flexibility for information interoperability.*
4. *Detailed analysis on most of the supporting areas for information interoperability such as security, transaction management, replication management, history tracking, data dictionary and indexing facilities.*
5. *Highlight different areas required that contribute to the research by presenting most of the past supporting services that require significant research to be undertaken.*
6. *Designing the access control system that links the different distributed users to the heterogeneous distributed information sources that is considered as the bond between the information sources.*
7. *Partially implement the proposed algorithms for the Computer Services Directorate, [State of Bahrain].*

1.4 Outlines of the Thesis

The remainder of this thesis is organized as follows: Chapter 2 provides a detailed explanation of related research in the area and an initiation of a personnel perspective of the

proposed prototype. Chapter 3 gives a study of various behaviors of the known database schemas. Chapter 4 gives a perspective on the database interoperability and discussion of the various requirements towards the achievement of the database interoperability. Chapter 5 discusses the cooperation behavior of the heterogeneous information sources. Chapter 6 is on the various issues to be considered when designing an interoperable information source. Chapters 7, 8 and 9 are concerned with the architecture, the design and the various supporting services of the proposed prototype respectively. Chapter 10 is mainly concerned with the various security advances considered by the proposed prototype design. Additionally, Chapter 11 provides a discussion on the design of the proposed prototype. Chapter 12 discusses the various interfaces of the proposed system. The conclusion to this thesis and future work are considered in Chapter 13.

Chapter 2

Related Research in the Area

The need for software to support data distribution has resulted in the development of the distributed database management software, which has itself been given impetus, by the rapid developments in telecommunications and network technologies. Distributed databases are a collection of multiple, logically interrelated databases that are distributed over a computer network, together with a distributed database management system which is a software system through which distributed databases are managed and through which the distribution is transparent to the user. Some distributed databases are homogeneous in nature, which means that the local database managers composed of a single DBMS product. Others are heterogeneous in nature, where different DBMS products make up the local data manager group [Simon95].

The most familiar names describing differences in Database Management Systems data models are "heterogeneous databases", "distributed databases", "multidatabase systems", and "federated databases". They relate to the problem of making different types of databases communicate and exchange data between each other in such a way keeping all performance, security, reliability, and simplicity to an acceptable level for the users. Systems will mainly provide a standard interface, which will make it possible for the databases to interoperate with each other. Systems such as Sybase provide an open interface, which includes commands such as "begin-transaction", "prepare-to-commit", "commit", and "abort". Similarly, Oracle provides an "explain" command which can be used to ask the system how it has optimized a given query. The information obtained this way can be used in global query optimization. Also, Ingres has released the open Ingres version, which is a distributed Information manager that allows a user to treat information resources within an organization as one unified information resource with the characteristics of a single local relational database. The data, which is accessed by Open Ingres in any local or remote databases, should be completely supported by Ingres. None of those is considered as the complete solution to the database interoperability problem. Early work on these topics indicates that they need to provide integration mechanisms for schema and process, while most of the work to date has concentrated only on schema integration.

The approach to cooperating databases should ensure that no bad effect on the other normal services offered by the database management systems and should support all the complementary services given by most of the existing will-known DBMSs, especially those related to constraints enforcement and owners' autonomy. Among those services are: schema translation management, programming language translation management, semantic inconsistency management, and other aspects related to the operating system and the communication protocol layers which are considered as outside the database research realm. Also, there are many other areas supporting the success of the database interoperation. Some of those areas are: dynamic access management, history tracking management, transaction management, data replication management, security and constraints enforcement management, data dictionary management, query decomposition management, and probably many other important services that are still unknown to research realms. Figure 2.1 shows a preliminary design of the interrelationship between the different services as seen at the present time. In Section 2.1 a thorough discussion of these different database services is given. In addition, the figure highlights the major areas of research in the database interoperability and the interrelationship between the different components.

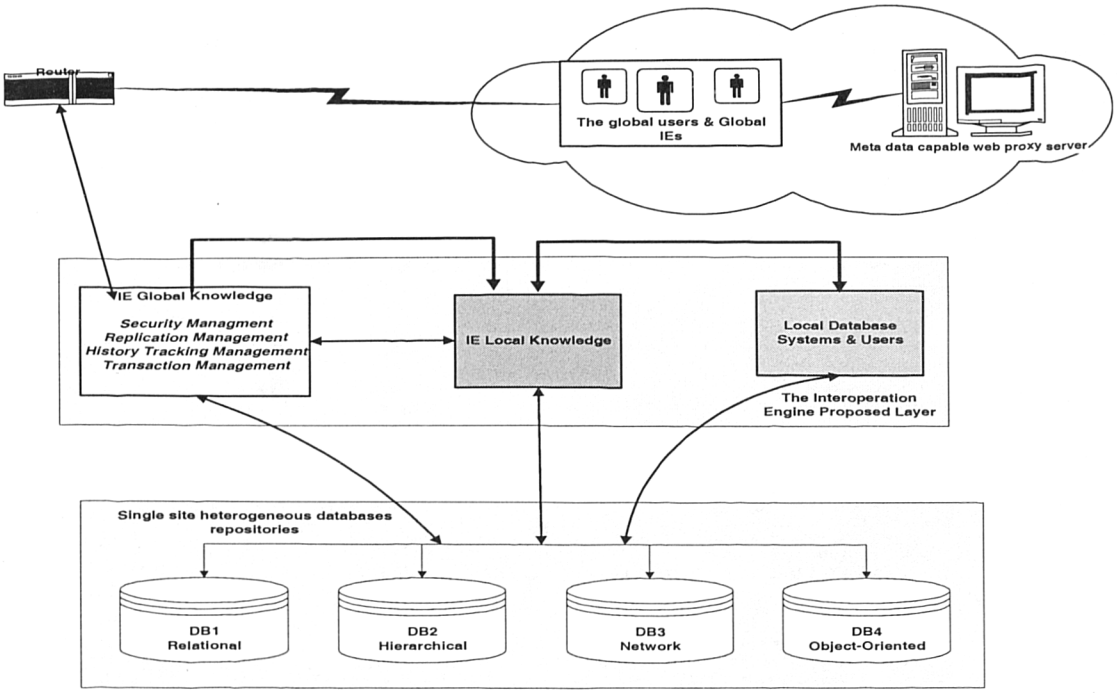


Figure 2.1

Although, the Meta Data Coalition Open Information Model MDC OIM version 1.1 proposals are also a close definition to the metadata registration of this research, the proposed prototype of this thesis could be considered as extension to the MDC OIM. This extension considers the metadata and data exchange mechanism, which forms the handling policies between both the information producers in one side and the information consumers on the other side. An interoperation interface that uses the metadata to link the different heterogeneous information sources together has been designed. Through the use of objects the cooperating information sources would dedicate objects responsible to transfer the records between the different scattered information sources.

Today Wrappers programs enables character or console based programs, which have difficulty running in a telnet or terminal session, to run in another platforms. Even though wrapping every scalar code would result in much lower performance than using the primitive types, but it is simple to implement.

The main objective of the database interoperability is to merge different schema type information sources together, which may requires consider different parts of the information source for the various types of users accessing it. Wrapping techniques will simplify the mission in the case of a complete schema to be used as if the users are working at the same machine the original databases are loaded on them. For this reason wrapping are unable to handle the various accessing users' requirements unless either creating multiple functions to meet the different requirements and wrap each of them individually by different name. Or on the other hand, define another layer having access to the different users requirements and the data sources breakdowns which is exactly the Interoperation Engine proposed layer.

An infrastructure capable of linking the heterogeneous distributed data sources willing to cooperate in an incremental manner required at this stage. This infrastructure should also be responsible for sending queries to all the cooperating sources asking for certain information of interest. It should deal with all the users in the globe having access to its local data sources.

Because of the diversity of information offered by the different sources and because each source must be queried in a certain way, problem of entering a single query and obtaining results from multiple sources is complicated. The proposed solution design tackles this point

in a technical manner. The mission of gripping information from multiple sources is done as a joint process between the information sources that will reply the query.

In general, most approaches to integrating heterogeneous information sources rely on another external form of representation usually based object models. Definition of relationships between the various objects is usually defined at an abstract and global levels rather than at the level of each resource, which the proposed layer is designed to achieve.

All those issues make a strong demand towards having a new dynamic mechanism for the cooperation process for the heterogeneous distributed data sources. This issue needs to be looked at in a new way. Although there is currently number of suggested static definitions for the cooperating data sources, no one considered the ideal solution for the rapidly increasing number of heterogeneous distributed data sources requires cooperation. Disco project [Tomasic96] is one example of the static definition of the necessary databases interoperation parameters and which this personnel view is considered, to an extent, as a dynamic implementation of parts of the Disco project.

A proposal for a middle engine based on number of hypothesis plus number of operations based on some prepositional standards for the proposed middle engine, which could act as an added-on facility on the Internet browsers, responsible for binding only these heterogeneous distributed data sources of interest has been discussed. This middle engine does not require the set of related applications to agree on one global view. The approach is based on the information availability and information demand. It is also depending on the information advertisement technique for the purpose of advertising an available information source. The infrastructure is also forming the foundation for all the supporting areas in the distributed databases to take place towards the interoperability support.

Works such as FINDIT [Busse94a] and DIOM, stands for Distributed Interoperable Object Model [Cardenas80], has addressed the problem of finding information in a large scale network of autonomous and heterogeneous databases and educating users about the available space of information. FINDIT mainly intended to advertise locally about the shareable local data sources. In this case, global users need first to send queries to remote information owners, and second the local user query will be sent in an indiscriminate manner to unknown database server. This is because not every database server knew about the available global information space it can access. The only available shareable information to the local site is the database servers linked directly with the local user server of that site. User servers are linked together so that they can exchange information. The local user query in this case may traverse to the outside world and may get rejected or the result of the query may not form any interest to the sender. By this it cost the user the time of sending the query till the answer is received.

Recently, substantial attention in the DBMS area has been attracted to the problem of heterogeneous database integration systems design. The practical importance of this direction consists at the development of tools making possible the coexistence of different DBMS supporting various data models, which meet applications requirements. The methodological importance of this direction arises from the fact that integration of heterogeneous databases requires methods of equivalent data model transformation and methods of constructing unifying data models and languages promoting generalization of various approaches to the development of DBMSs programming language [Kalinichenko90]. Because of this, a number of projects have been established in this area such as the Jupiter System, which is a prototype for multidatabase interoperability. In this system, the majority of the work has been done on the mapping mechanism between the different schemas of the databases. The mapping of schema has been considered between autonomous and possibly some heterogeneous databases [Murphy94]. Also, there are other projects in this area such as IRO-DB and ESPIRIT III. They provide a method of integrating heterogeneous data sources from the design perspective and the data perspective. They allow for the integration of heterogeneous object-oriented and relational DBMSs [Busse94c].

For the purpose of distribution it has been suggested that three sub-layers be created in the ISO/OSI application layer to form the mechanism of distribution process is supplied by

[Gligor84]. These three sub-layers will be responsible for global data management, distributed transaction management, and the structured data transfer protocols respectively. Four approaches have been worked out from past suggestions. Those are the CSIN, which is the Chemical Substances Information Network project [Smith81]; the UCLA DBMS project, which uses the ER model as its global - conceptual model [Cardenas80]; the XNDN project, which uses the relational model as its global model [Keimbleton81]; and the Multibase project, which does not require any changes to be done to the local databases or their DBMSs [Smith81]. All of the four approaches mainly apply the recommendations of the first sub-layer of the suggested proposal which is the global data management leaving the other two sub-layers with relatively little or no research until now. Also, non have treated schema integration between the heterogeneous databases as a dynamic process, and all of them are defining the integration as a static bridges between the different databases.

Solutions such as the one defined by the Jupiter system [Siberschartz94] are intended to define two schemas: the participation schemas, and the export schemas, which are normally part of the local schema for the global usage. In this case, if the link between the local schema and the global schema is down, then this will not affect the global schema users but they may get old data. This is subject to the changes that may take place during the downtime between the local and the global site. Also Jupiter uses a single uniform database language for the global access, where the users should learn to use this language.

There are some other approaches written especially for newly developed object-oriented database interoperability and as an extension they also capable of importing and exporting data from and to other databases [Busse94]. The *Common Object Request Broker Architecture* CORBA standard has been designed for the newly designed applications, and is considered as a standard to let objects over a heterogeneous environment to talk to each other regardless of the communication protocol used in each side of the internetwork [Burleson94]. As indicated in the literature, the real problem concerns legacy systems. CORBA is considered as a significant step in satisfying the needs of applications, and it makes a substantial contribution to interoperability and provides a high-level view for developing and integrating applications. However, it is limited to the classical object model and lacks an explicit view of resources [Kulkarni94]. The real significance of CORBA specification is for application developers who want to build new client/server applications that will work across disparate platforms.

Also, security adapted by CORBA is still a question. In the start, all the security applied by CORBA is mainly securing the object. It is different from the known security which is applied in databases, such as a particular person is allowed to view, update, or delete data, or as a person may be allowed to run some programs. All the security in CORBA is dependent on the power of the objects themselves and their behavior. It is been assumed that the object will not exceed its behavior. No doubt CORBA will facilitate linking the three-tier architecture data sources in a transparent manner to users. The Orbix product is CORBA compliant software enables users to downsize mainframe applications on Unix boxes and NT based networks.

CORBA is considered by many workers as the future standard distributed computing architecture. It only supports object-to-object interoperability and does not directly support distributed data or distributed transactions. It allows several heterogeneous object-oriented systems to co-operate and provides interoperability between applications on different machines by interconnecting multiple object systems through a remote object invocation mechanism. It would do so by means of what are termed object services (i.e. a DDBMS service and a transaction service attached via a gateway to the architecture.) IDL of CORBA stands for interface definition language. IDL specified methods can be written in and invoked from any language that provides CORBA bindings (currently C, C++, Smalltalk.) For one CORBA object to request something from another object, it must know the target object interface. The CORBA interface repository contains the meta data that lets components discover each other dynamically at run time.

Representative projects in the area of integration of heterogeneous information sources [Wiederhold92, Wiederhold93] include TSIMISS [Garcia94, Papakonstantinou95], which introduces an object exchange model and a specialized query language for integrated

information access. Other projects include work at USC [Macleod93], HP Labs [Shan93], and IBM Almaden Research Center [Carey94]. The entire area of distributed object management [Ozsu93] has also important impact on this research.

In addition, much of the work in the heterogeneous databases interoperability are also relevant [Tomasic96, Liu95, Sheth90]. However, our viewpoint is somewhat different from the previous research on the integration of the heterogeneous databases. Instead it is looked at the interoperability of databases from the perspective of making the interoperating data sources understand each other in terms of the database structure. The cooperating information sources would only create a view that can be understood by everybody in the cooperation process. All the created views will be governed by access rights system on the site where it resides for all the permitted users on that view.

Instead of applying static interoperation as applied by most of the existing solutions, which may affect the local autonomy, the full autonomy on the information source is given to the owner to decide about what parts of his data to be in the interoperation and who should use this data. The advances in the Internet could be utilized to propose a prototype to act as the basement for the database metadata management in a way similar to the existing HTML based search engines.

The current provided solutions requires human studies samples of the data, and determines the procedure to follow towards making use of the provided information space. In contrast to this, the IE intended to provide all the users with a unified interface on the available information space they can access. Users can chose the information parts that matches their requirements, merge them to their own information space, or only use them as a standalone information source.

2.1 Database Management Systems Required Services

Integration of data from heterogeneous data sources is an important aspect of database interoperability. When different database schemas are to be accessed from remote sites, the problem of schema and data inconsistency is seen as a major requirement. Along with the integration process there should exist some other assistance services to retain the reliability, the accuracy and the maintainability of the interoperation. The connectivity of the different components required in this regard was shown in Figure 2.1. The following sections are further explanation of each of the required components forming our vision for the database interoperation problem. It is an evaluation of all the requirements to interoperate between the different existing DBMS schema types. A review of the existing solutions to different parts of the whole problem has also been presented. A critical discussion to the current contribution needed in the area is discussed. Also, the major required features should exist in the future distributed databases have been defined. The interrelationship between the different interoperability components presented and discussed. A highlight to the different areas requires the research realm contribution also been presented.

2.1.1 Dynamic schema translation manager

The dynamic schema translation task is considered as the core problem of the database interoperability. Existing schema integrators read the metadata stored in a database and translate them into another ready schema. This will (first) duplicate the same information into two different schemas, (second) requires the receiving schema to be ready prior to translation. What is really required for interoperability of the heterogeneous databases is a dynamic engine capable of translating certain schema to other schema type without integrating the actual database records in another table. The migration size of the actual records between the different schemas should only be done according to the request. The dynamic schema translation manager should read any schema and transparently translate it to any other schema type for the global users. The proposed prototype is assumed to take care of the schema translation process in a transparent manner without intervening in the actual design of the database. So, it becomes evident that to achieve effective interoperation of remote, heterogeneous DBMSs, users must have a uniform, integrated access to the different DBMSs. The fundamental need for the uniform, integrated access to such databases arises because users cannot be expected to learn the use of many different DBMS and the

operational differences between them [Copmputer79, Gligor84]. The proposed prototype has taken care of the schema translation management in a way by reinventing the database in its non-normalized shape.

2.1.2 Transaction manager

Transaction managers [Verma95, Munakata97] where introduced to run classes of applications that could service thousands of clients. They do this by providing an environment that interjects itself between the remote clients and the server resources. By interjecting themselves between clients and servers, transaction managers can manage transactions, route them across different systems, load-balance their exception, and restart them after failures. The transaction manager can manage transactional resources on a single server or across multiple servers, and it can cooperate with other transaction managers in interoperation arrangements. For managing transactions within the information sources interoperation, for example, each user may have a certain level of importance on certain information sources defined into the transaction manager knowledge base. This manager is responsible of taking the definition from the knowledge base in the proposed prototype and grants the user the priority and access time to process his transactions according to the information given in the knowledge base. This manager should take care of the transaction flow in the interoperation and prioritize using, for example, criteria based on either the user level or database level to process the request. General-purpose transaction managers usually work in the kernel of the operating systems where the DBMSs are working under them and because of this reason they are not covered by the thesis in details. Rather referring to the required functionality by them in the database interoperability.

2.1.3 DBMS Programming Language Interpreter

The actual work of such interpreter is the responsibility of switching between the different DBMSs query programming language syntax [Burleson94]. As will be discussed later, the proposed prototype is designed so that the query interpretation is done on a simple non-normalized single table to simplify the interoperation process as much as possible to the end users i.e. information consumers. The interpretation process on the heterogeneous schemas is a lengthy and even complicated process if it is considered on the different schemas. It appears during the work in this thesis that part of the complete study is to design a compiler capable of understanding the different requirements of the heterogeneous schemas interoperation. Also, the interoperable compiler, as it may be called, should be enough capable of undertaking the diversion of the remote requests that may occur in the program source code. If the consideration is that users will be using the PL attached with their DBMSs then the remote data they will access should be part of their original database schemas. Translating between the different queries is considered as a tedious research that has not been yet undertaken and not the target of this research.

2.1.4 Query Decomposition Manager

Query decomposition processes [Kulkarni94] are the ones usually responsible to break down the incoming and the outgoing queries for the purpose to simplify the process. Of course, such process would speedup the query fetching operations. In case an ambiguous query is given to any of the heterogeneous schemas in the cooperation this manager will resubmit the query after rewriting it in a better manner. Query decomposition is a difficult task mentioned in the database research realm. Such process is normally activated after the query is first translated to the called data source query language. The proposed prototype handles the decomposition in a transparent manner by which all the queries are submitted to a single database table as will be explained later in chapters 7 and 8.

2.1.5 Data Dictionary

In addition to its normal rules, in the proposed prototype, it is responsible of clarifying the meanings of the local data definitions to the global information consumers. This would help the different users in matching between the different naming on the different distributed sites. This subsystem will cooperate in solving the problem of semantic inconsistency [Bouguettaya95], and will make users queries to reflect the actual needs as much as possible.

The data dictionary plays important rule in the database systems life cycle. It is considered as more general software utility than a catalog. Data dictionary systems are used to maintain

information on system hardware, system software, documentation and users, as well as other information relevant to system administration. A thorough discussion of the proposed data dictionary design is presented in Section 9.1.

2.1.6 Security Manager

The security manager mainly makes benefit of the treelike shape of the gathered metadata and users in accessing the information sources. The security manager is the part responsible for the entire schema types security issues. Access to the different schemas would be mainly done through this interface. Chapter 10 provides a detailed study on the IE unified security manager design.

2.1.7 History tracking manager

The history tracking manager subsystem [Murthy, Miura95] is a knowledge base responsible for keeping track of the status of the operations takes place in the local site. It should be also responsible for continuing the suspended operations handled by the transaction manager. It is in a position to be capable of giving the history about the sites where certain database operations are applied on them. Such services are handled in a transparent manner to the users. If certain critical operation fails this subsystem should be capable of exactly mentioning where the failure was and forward it to the transaction manager, which will be responsible for the continuation of the suspended processes. The schemas where indexes are used, the benefit of the history tracking will be much more than if the schema does not support indexing technique such as in the legacy hierarchical and network schemas where record fetching process is done similar to the linked list. This topic will be discussed in Section 9.2.

2.1.8 Replication Manager

Replication managers [Simon95] are those capable of liaising with other sites to get permission where a backup data store can be created for certain critical databases. This part should play an important role when the main site is down by transferring all processes to the backup site, and when the fault is over it should be capable of restoring all the changes to the master database and convert the process back to the original site. Also, it would play an important rule in the load balancing and data mirroring techniques. More discussion about replication and how it is handled by the IE proposal is given in Section 9.3.

2.1.9 Dynamic Access Manager

Dynamic access managers are mainly responsible for user ID's and passwords over the distributed sites and their attached database [Brodie95]. For our setup, this manager will also be liaising with other sites to authenticate and authorize users through the distributed databases. Because access to such data is a critical process and users have to access a strong authentication process then assigning such process is a strong recommendation. The process recommended by [Grimsom95, Cardenas80] can cooperate in security constraints on the distributed systems.

Requests are supplied to the *dynamic access manger,* which is the first contact point with every local IE, of course, after crossing the wide area links and the local operating systems layers. The dynamic access manager is assumed to carry two types of prioritization on the information source. First, local data stores and their categorizations according to their importance so that some of them are of higher priority on the others (compare this process with the weighted fair queuing in routing). Second, prioritization could be based on the group where the user belongs. In this case if the user or the user group is of high importance then the request will get more priority over the other requests. A third type evolving from the two types is the combination of both types. This type, of course, will give more priority over the two individually.

2.1.10 Overview of the Operating systems & communication protocols layers

Those two layers will be responsible for resolving differences in operating systems and communication protocols involved with the distributed databases. This should be handled transparently without the sense of the users. A preliminary study on the requirements from the communication protocol layer has been conducted in appendix B. People, whom have the knowledge of the current operating systems and communication protocols they are using, can define the parameters of these two layers [Tanenbaum87]. Also, there is a strong demand for

a very strong cooperation between research in the area of the distributed operating systems and the distributed databases in order to reach the required success in the interoperability area. Although operating systems area is considered out of the database research area, it will have a very strong contribution in the distributed database area. Some research areas are:

1. Changes in the names or locations of the stored databases involved in the interoperation should be reflected transparently to all the related sites at the time of change.
2. The distributed operating systems should liaise with the attached routers and should have briefing capabilities such as those exist in today's routers to keep users lists short, maintainable, and manageable.
3. It should give the full support required by the transaction manager described in section 3.2, or it may work on behalf of it.
4. It should contain a dispatcher to manage the movement of requests from local and global users. Also, It should consider all the users and their level of importance according to the priority list saved in a knowledge base.
5. It should consider multitasking only for large requests having low priority. This should involve cutting these requests into smaller parts and passing them one after the other. This should be done with the support of the query decomposition manager described in 2.1.4.
6. The OS is the part responsible for control access and protection of users. Authentication is the process where the user either gains access to the system or not. This is basically via a password or multiple passwords. This part of the operating system should be capable of handling the interoperability of all the required dynamic access needs to the distributed systems.
7. The difference in speed between storage and memory is an important factor not only for databases but also for all the operations, which requires swapping between memory and storage. As the speed of both becomes closer this will assist in solving most of the speed problems. Operating systems should be capable of creating its encryption mechanism and standards so that other responsible site should only know them. Although the encryption has disadvantages related to the speed, this can be negligible since the factor of the difference in speed between the memory and the storage becomes very close in today's systems as shown in Figure 2.2.

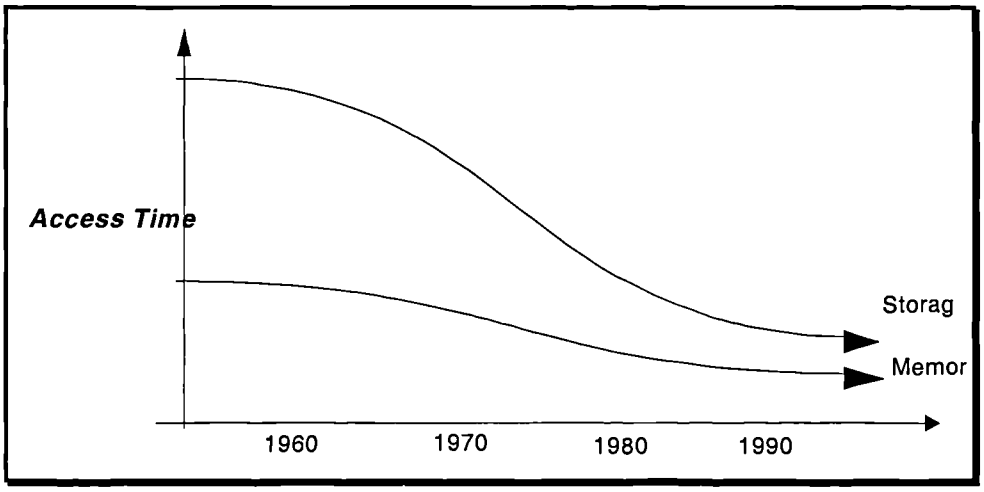


Figure 2.2

2.2 An overview of Multidatabase Systems

Computer applications in general, and databases in particular, are an integral part of the daily function of different groups of users and organizations. Databases in each of these environments have developed independently to meet specific requirements. Moreover, different DBMSs, which are usually incompatible with each other, have evolved to meet the varying needs in these independent environments. However, in today's networked world, separate autonomous data sources are no longer able to meet increasingly sophisticated user

needs. Related information important to the global application or request may exist in multiple, incompatible local databases. Users cannot be expected to manage system details of sending multiple requests in different languages (and possible different data models) to multiple information sources. Multidatabase systems provide integrated global access to autonomous, heterogeneous local databases in a distributed system. An important problem in the current multidatabase systems is the identification of the semantically similar data in the different local database [Bright94].

2.3 Database Schema Integration

The sources of data have been classified into three different classes. These are structured, unstructured, and semi-structured. Database in general, whether relational or the other schema types are considered as structured data sources. The web pages are examples of the unstructured data sources. The semi-structured data sources are the web pages with some known fields. This dissertation is only dealing with structured schema types in the experiments and for the future work all the schema types will be considered.

Heterogeneous database schema integration is defined as an approach to database design, application and management providing for the achievement of number of objectives. In most of the known systems these objectives where only partly achieved [Kalinichenko90]. The objectives are:

1. Joint usage of data from several heterogeneous databases as from a logically single database;
2. Multidatabase management (homogeneous presentation for an application program of a collection of various databases, maintenance of its integrity, provision of a common data manipulation language);
3. Data description and data manipulation language unification for various data models;
4. Maintenance of DBMS-independent generalized level of an application domain description;
5. Provision of application program independence of DBMS;
6. Continuous embarrassment of an extending spectrum of data representations and operations in computers.

Distributed databases provided the earliest solutions to information sharing in distributed environments [Alonso91]. It assumes that a single and integrated conceptual view of the databases must be provided to the users. Federated database schema and system architecture design have partially benefited from this effort. The design of both distributed and federated systems include functionality such as schema integration, query processing and transaction management. The federation database is just an extended technique of distributed databases. The main difference between them is that users of the distributed database systems access shared data only through the single conceptual schema as a centralized database. On the other hand, federated database systems support two classes: federation users manipulate shared distributed information through one or more federated schemes; and local users, to whom the federation is transparent, access local data only [Anderson93].

During recent years, there has been an increase not only in the automation of reasoning tasks, but also in the awareness that complex reasoning tasks need to be automated. Until recently, developments in the theory and practice of databases have proceeded largely on the assumption that users will consult a single appropriate knowledge base when they require information. This assumption is valid in many situations where knowledge/data are stored in a single knowledge base. Complex reasoning tasks on the other hand, may need to integrate information from a multitude of different sources. These sources may be databases (relational or otherwise), or knowledge bases (of logical or other forms) [Subrahamanian94].

The integration of existing databases provides for a uniform access to data stored by different database systems. Besides the integration of different data models and query languages, transaction management has to be provided. So atomic commitment for heterogeneous

database systems is needed. In homogeneous distributed systems, two phase commit is the most commonly used protocol. It uses a ready state for local transactions in order to wait for the global commit or abort decision. In heterogeneous systems, the use of a ready state is not feasible, since this requires all participating existing transaction managers to provide that state [Muth91].

2.4 Heterogeneous Database Systems

The provision of access heterogeneous distributed databases where each working under different environment becomes increasingly important in the most aspects of the daily life. It has been the subject of intensive research for at least a decade, yet the solutions published to date have addressed only part of the problem and have, in general, failed to provide an acceptable solution. Also it can be argued that the provision of easy access to heterogeneous distributed databases is the key problem in information systems today. The problem can be considered from the environment the databases working under them. Those are the different operating systems surrounding the databases, as well as, the different communication protocols used to transfer queries between distributed databases.

The distribution of databases is considered as one of the main overheads challenging the success of full interoperability among the databases. On the other hand, the heterogeneity of the environments is also considered as the second overhead, which challenges the success of database interoperability. It has been said that control of the distributed databases even if they are homogeneous and working under homogeneous environments is much more difficult than controlling heterogeneous and non-distributed databases. This is because distribution must take care of hardware problems such as differences in communication protocols, and software problems related to software compatibility such as operating systems and security issues.

Heterogeneity ranges from those differences in hardware to the differences in software. Hardware heterogeneity is the difference in networking topologies, data-links that represent the hardware interfaces between networks, and the physical communications, which represent the hardware connections between networks. On the other hand software heterogeneity is the one related to the differences in communication protocols, operating systems, and going to the lowest level of the DBMSs, which is the database schema representation. Both hardware and software in different systems has to adopt a standardization methodology so that communication between systems can be applied safely.

Data processing was adopted heavily by business during the 1950s and early 1960s, and large organizational data banks began to develop. At first, these where based on primitive file systems organized sequentially on magnetic tapes. In time, they progressed to indexed file systems on Direct Access Storage Devices (DASDs). In this way the term "database" became popular as a term for referring to all the data in any given domain.

The next attempt toward providing structured Data Base Management System "DBMS" came with the development of hierarchical DBMS, in the 1960s. The hierarchical model uses the concept of the record (or entity) as a collection of named fields to represent each individual element in the application domain.

The network model, which was refined in the 1971 by the CODASYL Data Base Task Group (DBTG), comes as a modification of the hierarchical model, which gives the freedom for any entity to be related to any other entity, those creating a network of related entities.

In 1967 Dr. F. Codd working as a research scientist for IBM in the USA, wrote a paper entitled "Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks," which analysis the use of Relational mathematics as a new technique for storing data. This was followed in 1970 by Codd's Landmark paper to the ACM *"A Relational Model of Data for Large Shared Data Banks"*. These papers and the work, which followed, caused the revolution of relational databases. Today, the majority of DBMS products are based on the relational model.

The entity-relationship (E-R) data model, which was refined in 1976 by P. Chen [Chen91], is based on a perception of a real word, which consists of a set of basic objects called entities and relationships among these objects. It was developed in order to facilitate database design by allowing the specification of an enterprise scheme. Such a scheme represents the overall logical structure of the database.

The next to come is the object-oriented model, which is accepted now by database programmers and is considered as an extension of the E-R model, and is based on object-oriented programming paradigm. This approach to programming was first introduced by the language Simula 67, which was designed for programming simulations. More recently, the language C++ and Smalltalk have become the most widely known object-oriented programming languages.

The Database Management System which is abbreviated as "DBMS" is the only term nowadays that carries the meaning of storing the data using one of the previous models and managing that data to get useful information. On the other hand, new DBMSs nowadays are using entirely different technologies than the relational data model, which is the most known model in the field of databases. This addition to existing data models gives some sort of new technology such as engineering design, and multimedia facilities such as pictures and sounds. By these new additions, tracking the old technology, which is in use, and adding the new facilities to them are very costly and difficult processes, and of course, time-consuming. This situation makes scientists and practitioners think of some sort of interoperability between those data models for the purpose of eliminating the gap in technology between those various types, and exchanging the new added technology in between them [Claude93].

Interoperability is required in three places: interoperability of process, interoperability of data, and interoperability in technology. On the other hand, interoperability of databases can be focused on three directions: heterogeneity of schemes, autonomy, which is distribution of control, and distribution of data. The next subsequent paragraphs are mainly talking about the different parts of heterogeneity faced by the databases.

A data model is a set of concepts that is used to describe the structure of the databases includes data types, data relationships, data semantics, operations, and consistency constraints. The various data models that have been proposed fall into three different groups: *object-based logical models*, *record-based logical models*, and *physical data models*.

Object-Based logical models are used in describing data at the conceptual and view levels, those, which links the external and internal levels of the databases. They are characterized by the fact that they provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. There are many different models and more are likely to come. Some of the more widely known ones are the entity-relationship model, the object-oriented model, the binary model, the semantic data model, the infological model, and the functional data model.

Record Based Logical models also used in describing data at the conceptual and view levels. In contrast to object-based data models, they are used both to specify the overall logical structure of the database and to provide a higher-level description of the implementation. Also they are so named because the database is structured in fixed format records of several types. Each record type defines a fixed number of fields, or attributes, and each field is usually of a fixed length. The reason behind keeping the record size for the models fixed is that, the insertion and deletion on the files of fixed length records are quite simple to implement, because the space made available by a deleted record is exactly the space needed to insert a record. If records of variable length are allowed in a file, this is no longer the case. An inserted record may not fit in the space left free by a deleted record or it may fill only part of that space. Some of the more widely known ones are the hierarchical model, the relational model, and the network model

Physical data models are used to describe data at the lowest level, which is the physical schema of the database. Those models are describing details of how data is stored, record formats, record orderings, and access paths. In contrast to logical data models, there are very

few physical data models in use today. Two of the widely known are unifying model, and frame memory.

Heterogeneity in networks means the differences in networking topologies such as those bus architecture networks, star, ring, mesh, etc. The heterogeneity in networking, illustrated in Figure 2.3, is measured by the differences in the networking protocols within each of the connected networks. This is considered as the real part affecting the performance of networks inter-connectivity in terms of packets sent between networks, and differences in packets bandwidths sent and received by each of the networks.

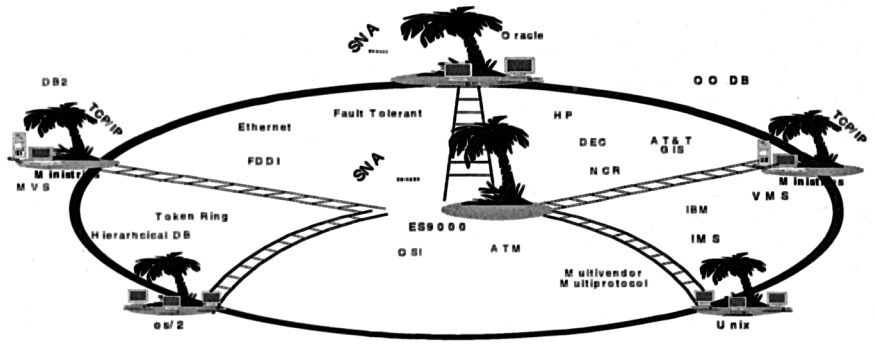


Figure 2.3

Networking protocols or as been some times called communication protocols are those set of conventions or rules used by a program or operating system to communicate between two or more endpoints. They are many different types, although they all allow information to be packaged, sent from a source point within one system, and delivered to a destination point in another system.

Communication protocols are those techniques ranging from single character-by-character transmissions with no error checking to those complex rules for moving large amounts of data among many devices. In general, communication protocols comprise three major areas: the method in which data is represented and coded; the method in which the codes are transmitted and received; and other non-standard information exchanges by which two devices establish control, detect failures or errors, and then initiate corrective actions.

There is often need to share data between systems purchased by any organization overall the world to support the needs of different business areas. Satisfaction of this need has been frustrated by an inability to provide the proper communication between the hardware and the software from different manufacturers overall the distributed networks. Sending requests across different distributed networks by using different communication protocols is a difficult peace of work; although it is a must based on an urgent need to it these days. Communication protocols on this situation have to be compatible and able to talk/exchange data between each other to handle the problems of differences in communication protocols.

So, exchanging data across different networking topologies throughout all the heterogeneous networks will be taken care by the routers, bridges, and hubs during the transmission process from point-to-point. So at this situation the network topology and type has no effect over the transmitted data overall the communication media sense the other networks know each node by its unique address.

For the purpose of sharing heterogeneous databases residing in or working under different operating systems the situation for such an environment is much different than if they are residing in a homogeneous operating systems environment. Functions handled by almost all the operating systems overall the world are related to controlling or managing the hardware, process management, scheduling or resource allocation, controlling the I/O access, memory management, file system services, support for hardware heterogeneity, and much more. The Data Base Management Systems need much more than the normal services of the operating

systems to handle much better, reliably sharable databases. It would need full transaction support with more complicated access methods. Of course, such methods are not similar to the normal file system services provided by operating system. Also they will need better memory and cache buffer management, and concurrency control [Ozsu93a, Jung97].

The normal existing services of current operating systems are not satisfactory to take over all the required database management tasks. This is especially true when it comes to the interoperation side of distributed and probably heterogeneous information sources. This is why the DBMSs are trying to adopt such services in their own areas, and such work is considered as an added drawback for the DBMSs [Ozsu93a].

The vision is to have the database management systems to interoperate between each other in a way to make any information source connectable with any regardless of where physically they are. For the purpose of simplicity the following scenario will clarify the situation; let's assume that three database pools trying to get information from each other to complete the requested process submitted by one of them. Suppose that the database pool 1 tries to get some information from tables residing in database pool 2, and database pool 2 also tries to get information from database pool 3 to complete the request of database pool 1, those can be considered as nested requests. Here it is assumed that each of the pools running a different operating system than others. Additionally, data sources are running under different communication protocols. By such situation the operating systems are facing some incompatibility factors, such as those related to the allocation tables, memory and catch buffer management, and others related to bandwidth fetching and handling. This example explains how the proposed system could deal with recursive-like requests.

2.5 Object-Oriented Analysis and Design

Analysis is the study of a problem, prior to taking some action. It means the process of extracting the needs of a system, what the system must do to satisfy the client, and not how the system will be implemented. The study of analysis comes as the result of software expansions and complexity. The problem of software expansions and complexity makes researchers and practitioners revise the problem-definition phase to add structure to the entire process to manage and control software development. Many practitioners came up with ideas and methodologies in analysis defined as the process of breaking down the problems to simplify them for the coding step. The most popular are those given by Yourdon and DeMarco in the structured systems analysis and design methodologies, while Fusion, OMT/Rumbaugh, Booch, CRC, and Objectory are the most successful object-oriented approaches.

As it has been said, "Necessity is the mother of invention", analysis and design methodologies are aimed at helping to simplify the way people think about problems. Object-oriented analysis and design methodologies complement the process of the previous analysis and design methodologies. They are consisting of mixture of the best steps applied in the previous analysis and design techniques. It is then said, objects better represent the world as people view it, rather than abstract the process to simplify the flow of data.

During the 1980's, hardware technology advanced considerably, which resulted in greater efficiencies and lower prices. By contrast, the software development process during the same period has been improved at a rate barely discernible. This situation has created a gap between the two technologies, leaving software designers unable to follow the speed and power of available hardware platforms. However, the development of object-oriented systems could place software development at the start of decreasing the gap in between the two technologies [Bertino93].

Analysis is using problem definition/modeling, while design focuses on solution specification/modeling. This means, systems design transforms the problem representation into a solution representation.

Systems analysis and design methodologies where introduced as the result of software expansion and complexity at the same time. During the analysis phase the problem is broken down into entities and relationship between these entities. The problem of software expansion

and complexity makes it necessary for researchers and practitioners to revise the problem-definition phase to add structure to the entire process to manage and control software development.

The object-oriented analysis and design methodologies have evolved from the structured systems analysis and design. Yourdon, who is the inventor of structured systems analysis and design, describes object-oriented analysis and design as an extension for the structured analysis and design methodologies.

Object-Oriented software development, including object-oriented analysis, object-oriented design, and object-oriented programming, is a promising approach to developing software systems in order to reduce costs and increase flexibility in general during analysis and design phases by allowing reusability, extendibility, maintainability, and programming in large [Luker94]. Although alone it will not eliminate all the analysis problems (and hard work and dedication are still needed to produce the best and most efficient software possible) it is the most promising. Many software-engineering researchers have proposed an object-oriented approach to software design, because they see that this approach generally imitates reality better than traditional data flow or state transition design approaches. By this approach, which is the same in all the analysis and design approaches, the problem is broken down into entities and communications among these entities. The entities then conceptualized in a hierarchical manner to use the properties of inheritance. In many ways the analysis is the design in object-oriented analysis, once the problem is analyzed. Object-Oriented analysis technique involves modeling the process as seen by those who work with the system. This makes the object analysis process easier and straightforward than other techniques. Also there is another reason behind the success of object-oriented technique, which is the combination of the data description and operations performed on that data into one entity, which traditional analysis techniques provide separately. This enable objects to capture more information of greater importance about the process being modeled than virtually all other techniques [Coad91].

This section proposes an analysis and design technique for the IE [Ashir2000] and in the light of the study discusses systems, which benefit from the object-orientated techniques such as object-oriented operating systems, and object-oriented languages.

Analysis is the process of investigating how a particular business system currently operates, modeling the system and determining the essential characteristics of potential automated solutions. This often involves observing complex processes, interviewing people involved in the process and laying out the process as a data flow, state transition, or other pertinent modeling methods.

The main goal of the analysis phase is to build a problem model - that is, to create a description of just what exactly is needed. These may take the form of interviews, specifications as to level of performance [Yourdon93].

In the present time most of the analysis techniques in use are those based on data flow diagrams. Yourdon, Constantine, DeMarco, Page-Jones, and others have written about SA/SD. Ward and Mellor have added real-time extensions to the SA/SD. SA/SD is pervasive, applicable to many problems, and well documented. SA/SD supports three orthogonal views of the system: object, dynamic, and functional models. It stresses functional decomposition. A system is viewed primarily as providing one or more functions to the end user [Frank95].

In addition, endless demand for better software development methodology that reduce both overall systems development and maintenance has led to the acceptance of object-oriented analysis and design methodologies recently. As an overall process, object-orientation tends to use the normal human thinking in expressing problems. Also, the methodology is considered as a new toolkit that can be added to the traditional approaches (such as data flow, process flow, and state transition diagrams); and it is not in place to replace those traditional approaches. In recent days most client/server application development tools are emphases object-oriented features, because it is found to be very effective in business problems.

The boundary between object-oriented analysis and object-oriented design is not clearly defined in the literature. Some processes used by one author during analysis may be included in another author's design technique. Some authors said that object-oriented design could be considered as a superset of the object-oriented analysis phase. They define the object-oriented analysis as the one model the problem domain by identifying a set of semantic objects that interact and behave according to systems requirements. On the other hand, they define object-oriented design, which is suppose to be language independent to, as the part models the solution domain which includes the semantic classes and other classes defining (interfaces, applications base utilities, etc.) identified during the design process.

Many course syllabi and textbooks subscribe to the notation that object-orientation requires nothing more than a change in language. But it is considered as a true paradigm shift in software engineering. It requires a complete change of worldview [Luker94].

The most important issue to figure in object-oriented analysis and design is the identification of classes, attributes, and methods or it is also called behavior. The relationship between classes is also part of the identification process.

There are three categories of object-oriented analysis and design methodologies that can be figured from the existing methodologies: 1) process only, 2) representation only and 3) process and representation. The first is the procedural methods supporting object-oriented analysis or design and not including any object-oriented analysis and design diagrams or notations. The second refers to graphical notations or diagrams for depicting the output of object-oriented analysis and design and focus on visually representing a design, and not on how to derive a particular design. The third is encompasses both processes for performing object-oriented analysis and design and representations for specifying the results. Next, at this comparison of three object-oriented analysis and design methodologies only category three will be considered [Monarchi92].

After a preliminary study of the analysis and design methodologies the Object Oriented Modeling and Design with the Unified Modeling Language UML as the modeling language that will be followed as the standard modeling notation to create the different kind of flowcharts [Rumbaugh91, Fowler97, Booch99, Rumbaugh99] has been chosen. This selection has been made based on the fact of the availability of the different assistance documentation of methodology as well as the facilities that are available in the methodology that will facilitate the design process. Also, the quick understanding of the different modeling stages is behind choosing the methodology. This part of the thesis, which is thoroughly discussed in appendix C, was not aiming to compare the different analysis and design methodologies rather it aims to select the most appropriate analysis and design methodology that fits the different stages requirements of the proposed prototype.

2.6 Object-Oriented Databases

Object-oriented database management systems are considered as the rich type systems that supporting user defined abstract types. Also, they are supporting more complex object structures with nested objects and richer languages, which overcome the well-known impedance mismatch problem. The relational model, complete objects cannot be defined or modeled directly, because relational model does not support the notion of inheritance. Contrary to the relational model, there is no universally specified object model. There are number of features that are common to most model specifications, but the exact semantics of these features are different in each model.

An object-oriented database inherits all the facilities provided by the object-oriented analysis and design and programming and the supported object-oriented languages. The main strength of this type of DBMS is the usage of the new programming techniques provided by the object-orientation by which the most are inheritance and encapsulation.

Storage techniques for object oriented database management systems proposed up to date stay between two main approaches known as the direct model and the normalized model. In the direct model, which is the same as the one used in relational database management

systems, objects are stored in the same way in which they are defined in the conceptual schema; that is, the unit of storage is the same as the semantic unit. More specifically, objects belongs to the same class are stored in the same file and each file record is an object instance of the class. The high transfer rate of complete objects her is considered as an advantage. On the contrast if access is to be done on some of the attributes in the object it needs lot of process in the sense the field sizes are not stable. Also, it is difficult to add new attributes to the existing object unless spare attributes already considered at the end of the object. Additionally, if the majority of the existing attributes are null value these can be considered as a waste of space. In the normalized model, the objects are decomposed into atomic components stored in different files. The relation between the various components is maintained by means of object IDs.

A hybrid approach between the direct model and the normalized model can be adopted. Complex objects are decomposed, but components are grouped together according to access patterns and the components that are accessed contemporaneously are stored in the same file. Although this approach is very efficient it requires having prior knowledge about the exact access pattern for the classes that are equivalent to the tables in the relational databases. The proposed interoperation layer IE has considered such access pattern information. The assumption here is to have a direct model similar to the relational model of storage that is capable of using the OID to handle the different objects stored in the different files. Also, for easier manageability, the similar attributes will be of similar sizes.

The saving in the storage usage and the accompanied rapid information retrieval of the object-oriented databases are a well-known advantage behind the strength of this technology. On the other hand such advantages will create difficulties in the manageability of the information stored using such technique. The object-oriented databases use a technique known as the property list to trace the places of each of the object attributes. This technique has information such as an identifier for each attribute of each individual object that is different from the OID followed by the attribute information size to be used and the value.

Chapter 3

Close Study in the Heterogeneous Databases Schemas

As known from the literature, the international definition of the primary goals of the database systems is to provide an environment that is both convenient and efficient to use in retrieving and storing database information. The basement for such requirement is the database schema that is derived by the business rules and the different constraint assignments.

Database schemas almost look the same. The only difference between them is in the way the different levels related to each other. For example, the hierarchical schemas always having the relationship between the different record types is either a one-to-many or a many-to-one between the adjacent levels, which means lateral links between levels or files are not allowed as shown in Figure 3.1(a). On the other hand, the difference in the relationship between the adjacent levels of the network schema is arbitrary, which means that the relationship can also defined as many-to-many as shown in Figure 3.1(b). The relationship on the relational schema in between the different levels can take the shape as many-to-many, many-to-one, one-to-many, and one-to-one at the same time. Also the main difference in the relationship between the different levels is that it can be done between any of the level. This is because the definition of the different levels is represented in the form of files as shown in Figure 3.1(c).

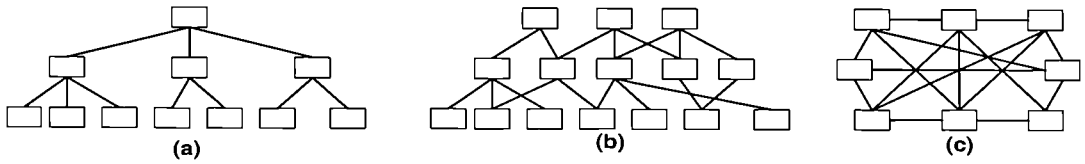


Figure 3.1

In general, the enhancements applied to the database schemas are the same to those applied to the networks connectivity overall the world. The relational schema is the one equivalent to the fully meshed network. In such networks if the link from one side is inactive it could be activated from another side, which is the same situation if a relationship between two tables is corrupted from one side than still it could be built from the other table. If the relational schema is compared with the non-relational schema, it will be known that using such schemas is more secure than using other non-relational schemas in the sense they are the most rebuild-able schema types.

The overall objective of this chapter is to present the most common approaches on DBMSs and give, while trying to put them into the correct context of history. The different sub objectives of this chapter are to thoroughly study the behavior of the different schema types, highlight the differences and similarities while discussing each of the schema types, discuss the different scenarios of schemas amalgamation techniques and design the necessary algorithms that make the schema. Additionally, a highlight to the object-relational DBMSs will be discussed with highlighting the most important advantages of such technology advances.

3.1 The Relational Schema

Edgar F. Codd first introduced the mathematical relational model in [Codd70] and later in number of papers [Codd71a, Codd71b, Codd71c, Code72]. It is representing the database as

a collection of relations. Informally, each relation represents a simple table or simple file. Each relation also consists of at least one attribute, which is equivalent to the field name in a table. The relation also consists of tuples, which forms the records in the table. Most of the relational schema engines will reserve space for the full record size regardless of how much data has been entered of the record. The relational engine will provide dynamic pointers between the attributes and between tuples in a relation. Also dynamic pointers are established in the time of binding between the relations. Dynamic binding is considered as a drawback when the process speed was low, but with the improvement in the process speed this drawback has become negligible. Of course, the statically defined pointers are considered as an advantage over the dynamically defined pointers, especially in retrieving operations when having enormous number of records in the table.

The relational model differs from the network and the hierarchical models. It does not use pointers within the database management system programming language. Instead, the relational model relates records by the values they contain in both sides of the tables, which are internally predefined by the relational engine. This freedom from the use of pointers allows a formal mathematical foundation to be defined. This is why the network model and the hierarchical model are tied more closely to the underlying implementation of the database than is the relational model.

The relational schema should not be conflicted with the nested relational data model. The nested relational data model is a non-normalized data model. It is considered as a non-first normal form relational data model. In this model all the data is represented as a single table. As an example, the representation of a department schema which almost looks like a spreadsheet as displayed in Figure 3.2.

Department Schema

			Employees			Projects		Locations	
Number	Name	Manager	Name	Department	Age	Name	Location	Department	Location

Figure 3.2

In the relational model, complex objects cannot be defined or modeled directly, because relational model does not support the notion of inheritance. Such feature only supported by the object-oriented databases. Although such facility could be met by using the relational schema, but this is considered as an additional work compared with the object-oriented database features, which are thoroughly explained in Section 3.4.

3.2 The Hierarchical Schema

Hierarchical database management systems became commercial available in the late 1960s with IBM IMS and the DL/1 language. The hierarchical databases have a treelike structure with every node of the tree representing different tabular file. Hierarchical databases are not binary trees. Each file is related to one another through the link to the record above or below. Hierarchical databases do not allow lateral links between files. Also, in the hierarchical model an owner record or file can have many members, but the member records can only be owned by one owner record. A treelike structure is a root node with sub-trees $t_1, t_2, \ldots, t_n$ that are themselves trees and have no nodes in common.

A hierarchical database consists of a collection of records that are connected to each other through links. The record is similar to the network model and the relational database model. Each record is a collection of fields, each of which contains only one data value. A link is an association between precisely two records. A link here is similar to the link in the network model. Links in the hierarchical and network databases are considered as a static links, while in the relational databases they are considered as dynamic links. Links in the relational databases for the different relations are established when required by applying relational algebra equations.

Consider a database that represents the library reservation and searching database in a university. There are four record types: subject information, publisher information, author

information, and book information. All the sub-trees in the hierarchy are assumed to be in ascending order. A sample relationship database appears in Figure 3.3 for the four level hierarchical database.

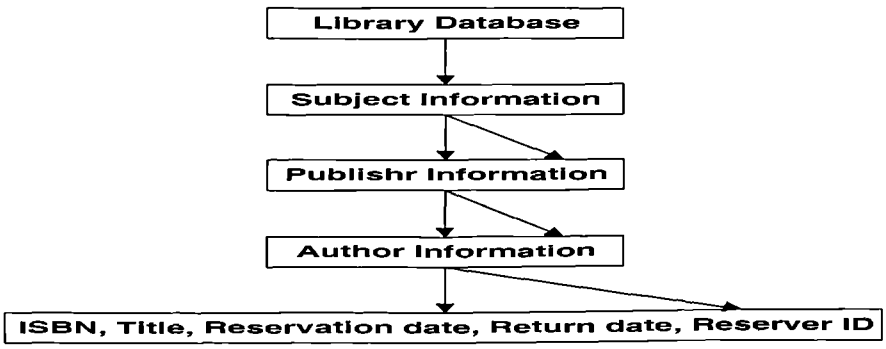


Figure 3.3

The most basic hierarchical schema representations are those, which accept duplications in records in some of their tree levels. Such hierarchical databases are very basic and not using virtual parent-child relationships VPCR. The VPCR is the representation for the hierarchical schemas where two separated schemas could be linked. The VPCR technique considered as an overcome to the problems of M: N relationship, the case where a record type participates as child in more than one PCR type and N-arry relationships with more than two participating record types. Also, a dynamical VPCR is useful in establishing relationship operations between the hierarchical model and the other models. For the purpose of this thesis the simulation will use the simplest hierarchical schema representations and will discuss the possibilities of simulating the most complicated hierarchical schemas databases.

The nature of the hierarchical databases differs from the nature of the relational files. The hierarchical files are those trees with root and branches. While in relational databases each record or file has a set of attributes and a set of entities that forms the table. Separate tables relate to one another through a common attribute. Any attribute in the table used as a key link with the other tables. In the case of the hierarchical files no relationship between the different records is defined other than the links established by the hierarchy itself. Only one-to-many and one-to-one relationships are directly represented in the hierarchical model. The many-to-many relationship is much more complicated. Each level of the hierarchy in the hierarchical database considered as a table in the relational database. The only way to set a relationship between a hierarchical database and a relational database is to search for the required relationship key from the hierarchical database or it can be done through using VPCRs. After finding the related record, links to other hierarchy levels traced through the static links created by the hierarchical schema. For the example given in Figure 3.3, the relationship key between the library subsystem, which is a hierarchical database, and the student registration subsystem, which is a relational database, is the personnel number of the persons. In this case if any student reserve a book then his number will be recorded into the hierarchical database, which forms the relationship pointer between the two schemas.

Hierarchical schema may consists of a number of hierarchical schemas. Each hierarchical schema or hierarchy consists of number of record types and parent-child relationships. Parent-child relationships are those links represented as pointers between the different levels of the hierarchy. Figure 3.4 shows a simple hierarchical diagram for a hierarchical schema with six record types and five parent-child types.

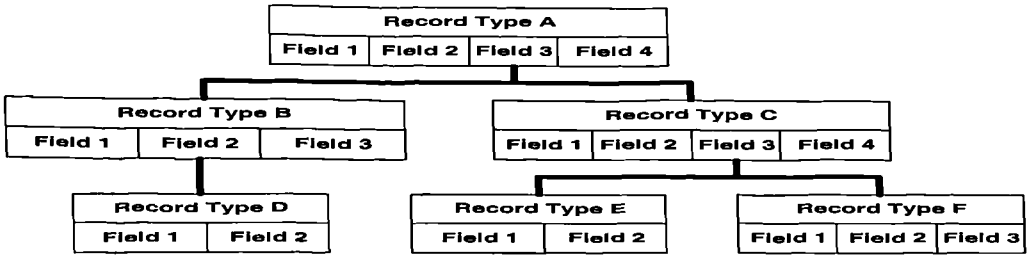


Figure 3.4

Simulating the hierarchical schema of figure 3.4 will involve defining all the necessary links between all the levels of the hierarchy. Also, the schema should contain pointers between the records in any of the levels and between the different levels. The following seven record types are representing the hierarchy of Figure 3.4.

Start Pointers (Record Type A Head, Record Type A Tail)
Record Type A (Field 1, Field 2, Field 3, Field 4, Successor Record Type A, Predecessor Record Type A, Record Type B Head, Record Type B Tail, Record Type C Head, Record Type C Tail)
Record Type B (Field 1, Field 2, Field 3, Successor Record Type B, Predecessor Record Type B, Record Type D Head, Record Type D Tail, Backward Record Type A)
Record Type D (Field 1, Field 2, Successor Record Type D, Predecessor Record Type D, Backward Record Type B)
Record Type C (Field 1, Field 2, Field 3, Field 4, Successor Record Type C, Predecessor Record Type C, Record Type E Head, Record Type E Tail, Record Type F Head, Record Type F Tail, Backward Record Type A)
Record Type E (Field 1, Field 2, Successor Record Type E, Predecessor Record Type E, Backward Record Type C)
Record Type F (Field 1, Field 2, Field 3, Successor Record Type F, Predecessor Record Type F, Backward Record Type C)

The representation for the record type in the hierarchical structure using structured programming languages such as C or Pascal will looks like the code shown in Figure 3.5.

```
structure type {
                type variable 1 ;
                type variable 2 ;
                .
                .
                .
                type variable n ;
} ;
```

Figure 3.5

For the hierarchical data type representation, this will involve creating all the record types as defined in the structure of Figure 3.5. After creating all the structures, the necessary links between them must be created. This step will involve defining the owner of the structure and the members for that structure. For the purpose of representing the hierarchical schema, the definition on the owner side will be defined, as the side has no duplication. On the other hand, the member side should accept duplication to form the many side of the relationship.

The treelike structured files came in number of shapes. The simplest are those balanced and binary trees. Balanced if the number of levels is equal or nearly equal for all branches of the trees. Given that record requests are uniformly distributed, a balanced tree will require on the average the least number of block accesses to locate a record. On the other hand, the binary trees are those where each record contains only one value and two branches. Such representation is of interest for two reasons. The first is, the structure is appropriate for memory because of its simplicity, and hence is also useful within a block of file storage. The second reason is that the behavior of dynamically changing binary trees has been investigated both theoretically and statistically.

In general the algorithm displayed in Figure 3.6 is used for obtaining records serially from a tree. The proposed algorithm can find records either in ascending or descending orders. Figure 3.7 explains how the records in any of the levels are to be sorted. Later in this thesis

supporting algorithms will explain the maintenance of the flow of the records in case of addition and deletion of record within the sub-trees.

```
proceed :       go to the left sub-tree,
          if not terminal block
                then proceed,
          else
                process the records of the terminal block,
                go up to the root for this sub-tree,
                if there is a record,
                      then do ;       process the record,
                                      go to the next sub-tree,
                                      proceed,
                                      end;
                      else do ;
                                      if this is the root for the entire file,
                                      then done,
                      else
                                      proceed,
          end;
done :
```

Figure 3.6

Internally hierarchical database uses the properties provided by the linked list. Those properties consist of preparing the linked list for sorting operations, such as sorting the list in ascending and descending orders. Figure 3.6 shows how the links are provided between the different records in the hierarchical databases. There are two types of links provided in the hierarchical databases. The first are the vertical links which are the links making the hierarchical structure building of the tree. Those are basically linking the levels of the tree. The second are the horizontal links, which are the links forming the alphabetical or the numerical order of the different records of a single level of the tree. Making use of the two links the movement within the tree is shown in Figure 3.7.

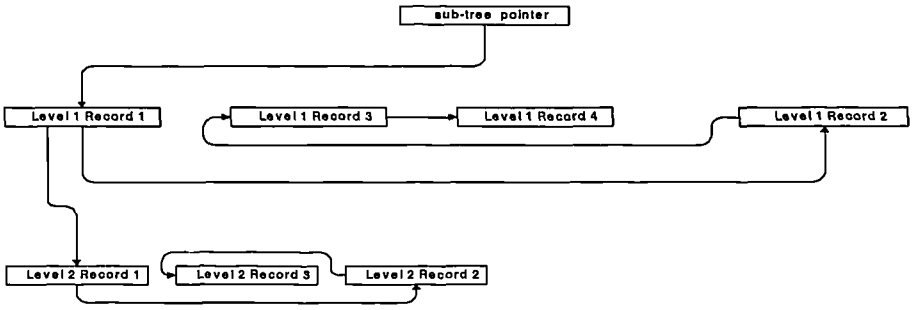


Figure 3.7

As the example for representing the hierarchical schema in Figure 3.4. Tables representing the example data simulated to work as a hierarchical database are given in Figure 3.8. All the supporting algorithms working with those five tables are depending on the traversing algorithms written to simulate the hierarchical database schema.

The hierarchical model is usually implemented by using hierarchical files, which preserve the hierarchical sequence of the database. In addition, various options including hashing, indexing, and pointers are available, so efficient access to individual records and to related records can be gained. Most hierarchical systems provide many such options for tuning the performance of a database system.

Head

SubjectHead
1

Subjects

Address	Subject	OtherInfo	NextSubject	PublisherHead
1	Computer	"	2	1
2	Sports	"	3	6
3	Physics	"	0	4

Publishers

Address	Publisher	OtherInfo	NextPublshr	AuthorHead
1	McGRAW	"	2	4
2	Osborn	"	0	1
3	PWS-Kent	"	0	0
4	ACM	"	3	7
5	IEEE	"	0	0
6	PrintcHal	"	7	0
7	JohnWiley	"	5	8

Authors

Address	Author	OtherInfo	NextAuthor	BookHead
1	Allen	"	2	6
2	Ambel	"	3	7
3	Ammon	"	0	8
4	Barbara	"	5	1
5	Bell	"	6	2
6	Bradley	"	0	4
7	Ceri	"	0	9
8	Charles	"	9	10
9	Cheng	"	10	11
10	Davis	"	0	12

Books

Address	ISBN	Title	ReservationDate	ReturnDate	ReserverCPR	NextBook
1		ABC Comp	"			0
2		Learning C	"			3
3		Visual Bas	"			0
4		Visual C++	"			0
5		Internet Di	"			5
6		Master Inte	"			0
7		Windows	"			0
8		Using Wor	"			0
9		Building A	"			0
10		Simple Elic	"			0
11		Elementary	"			0
12		World Cup	"			0

Figure 3.8

When representing the data of Figure 3.8 using the pointers defined in all the five tables starting with "Head" record, which defines the "Subject Head" and ending with "Next Book" pointer, it will involve visiting all the defined pointers as in the linked lists to form the tree shown in Figure 3.9.

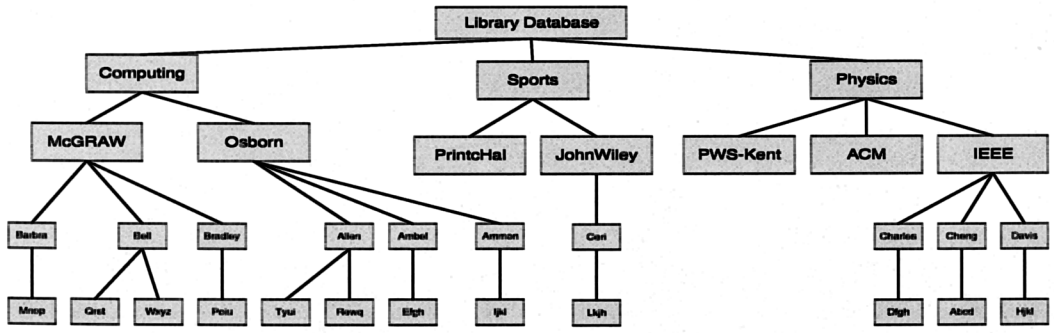


Figure 3.9

As an example reading Head will give the Subject Head as record number 1 in the table "Subjects" which is Computer. From this record the Publisher Head will be taken which is in this example record number 1. Visiting the Publishers table will again give the link to the Authors table. These processes will only end up when the "Next Subject" in table "Subjects" is 0, which means there are no other subjects defined. This process complies with the contents of the algorithm of Figure 3.6. Figure 3.10 explains the complete algorithm of traversing the data defined in figure 3.8 tables which forms the tree of Figure 3.9.

```
Hierarchical database traverse algorithm
        Make all the databases  ready in the open mode

        Locate the pointer to the first record in the root record types of the hierarchy
        Do while not end of the linked list in the highest record set
                If the required record then
                        Apply the required operation on that record
                        Get the head pointer of the next level
                Else
                Do while not end of the linked list in the next child record set
                        If the required record then
                                Apply the required operation on that record
                                Get the head pointer of the next level
                        Else
                                .
                                .
                                .
                                                Point to the next record in the lowest hierarchy data set
                                        End of lowest record set in the hierarchy

                        Point to the next record in the second hierarchy data set
                End of second level record set in the hierarchy

                Point to the next record in the root hierarchy data set
        End of the root record set in the hierarchy

        Close all the opened data sets
End of the hierarchical database traverse algorithm
```

Figure 3.10

As one of the most important rules for dealing with hierarchical databases is to keep the continuity in the flow of the linkage between the records of the different levels in such databases. A hierarchical database does not allow disconnection in the linkage between any two adjacent levels of the structure. If such situation occurs then there is no way to get all the related information to such disconnected records. Assume a hierarchy with four levels as illustrated in Figure 3.11. The data entry user should be able in this case to enter only the subject record type. Also, entering the subject and the publisher data is another acceptable case. The following table explains all the necessary cases and whether it is acceptable case or not when entering new records in the hierarchical database of Figure 3.11.

In general, there are many feasible methods of designing a database using a hierarchical model. In many cases, performance considerations are the most important factors in choosing one hierarchical database schema over another.

Subject	*Publisher*	*Author*	*Title info*	*Result*
0	0	0	0	Rejected
0	0	0	1	Rejected
0	0	1	0	Rejected
0	0	1	1	Rejected
0	1	0	0	Rejected
0	1	0	1	Rejected
0	1	1	0	Rejected
0	1	1	1	Rejected
1	0	0	0	*Accepted*
1	0	0	1	Rejected
1	0	1	0	Rejected
1	0	1	1	Rejected
1	1	0	0	*Accepted*
1	1	0	1	Rejected
1	1	1	0	*Accepted*
1	1	1	1	*Accepted*

Note : *0* *Empty Field*
1 *Entered Field*

Figure 3.11

The many-to-many relationship in the hierarchical database can either be met by duplicating records, such as what happens in the past library example in between the publisher and the author information or by creating more than one hierarchy and establishing a PVCR between them. Figure 3.12 shows the hierarchical add algorithm listing.

```
Hierarchical database add algorithm
    Make all the data sets ready in the open mode
    If data entered are valid and keep the connectivity between the hierarchy levels
        Apply Hierarchical database write record algorithm
    Else
        Reject the data entry case
    End
        Close all the opened data sets
End of Hierarchical database add algorithm

Hierarchical database write record algorithm
    Is pointer exist for the first record in the first level hierarchy
    If exist
        Set indicator(1) "First level record set" first record = true
        Set indicator(2) "Previous record in the first level record set pointer" =0
        Do while first level data set not pointing to the end of the record set
            Point to the first record in this level (First)
            If the key field in this record is less than the entered dat
                Keep information of the previous record address
                Get the pointer to the next record from the current record
                Change indicator(2) = Next record address from the current record
                Change indicator(1) = false
            Else if the key field in this record is equal to the entered data
                From the first level record set get the pointer to the second level record set head pointer
                Set indicator(3) "Second level record set" first record = true
                Set indicator(4) "Previous record in the second level record set pointer" =0
                Do while second level data set not pointing to the end of the record set
                    Point to the first record in this level (Second)
                    If the key field in this record is less than the entered data
                        Keep information of the previous record address
                        Get the pointer to the next record from the current record
                        Change indicator(4) = Next record address from the current record
                        Change indicator(3) = false
                    Else if the key field in this record is equal to the entered data
                        From the second level record set get the pointer to the next level record set head pointer
                        Set indicator(3) "Second level record set" first record = true
                        Set indicator(4) "Previous record in the second level record set pointer" =0
                            .
                            .
                        Process the Nth level data set
                        If exist
                            Give message record is exist
                        Else
                            Check if indicator(N-1) = true
                            Change the pointer in the level before the last to point to the new record address
                            Only add a new record set in the lowest level of the hierarchy
                        Else (This mean the record is less than the last traced record and comes before it)
                            Adjust the pointer in the previous record to point to the new record
                            Add the new record and set the next record pointer to point the previous record next record pointer
                        End
                                .
                                .
                    Else
                        Check if indicator(2) = true
                        Change the pointer in the level before the last to point to the new record address
                        Only add a new record set in the lowest level of the hierarchy
                    Else (This mean the record is less than the lat traced record and comes before it)
                        Adjust the pointer in the previous record to point to the new record
                        Add the new record and set the next record pointer to point the previous record next record pointer
                        Add the next levels records
                    End
                Else
                    Check if indicator(1) = true
                    Change the pointer in the level before the last to point to the new record address
                    Only add a new record set in the lowest level of the hierarchy
                Else (This mean the record is less than the last traced record and comes before it)
```

Figure 3.12

```
                Adjust the pointer in the previous record to point to the new record
                Add the new record and set the next record pointer to point the previous record next record pointer
                Add the next levels records
              End
          End of the second Do
      End of the first Do
   Otherwise
      Sets are empty, Add the first record in the sets
   End
   Close all the opened data sets
End of Hierarchical database write record algorithm
```

Cont. of Figure 3.12

The physical database representation refers to the hierarchy that is actually stored. As an example, in the IMS this is represented in the form of a physical Data Base Definition DBD using the DL/1 language. The following figure 3.13 shows the definition of part of a physical database that corresponds to the hierarchy shown in figure 3.4.

```
1 DBD NAME = OF FIGURE 5
2 SEGM NAME = RECORD TYPE A, BYTES = 40
3 FIELD NAME = FIELD 1,   BYTES = 10, START = 1
4 FIELD NAME = (FIELD 2, SEQ), BYTES = 5, START = 11
5 FIELD NAME = FIELD 3,   BYTES = 15, START = 16
6 FIELD NAME = FIELD 4,   BYTES = 10, START = 31

7 SEGM NAME = RECORD TYPE B, PARENT = RECORD TYPE A, BYTES = 20
8 FIELD NAME = (FIELD 1, SEQ), BYTES = 5, START = 1
9 FIELD NAME = FIELD 2,   BYTES = 10, START = 6
10 FIELD NAME = FIELD 3, BYTES = 5, START = 16
.
.
N DBDGEN
N+1 FINISH
N+2 END
```

Figure 3.13

3.3 The Network Schema

The relationship between records in the network databases is always represented as many-to-many, which makes an arbitrary links between the different records. Also they can be one-to-many and many-to-one at the same time. The one-to-many relationship in the network data models known as a set type. This kind of data model allows the definition of system owned set, which is a set with no owner record type. Also, it allows the multimember sets, which are similar to those defined in Figure 3.4. In addition, it allows the definition of recursive sets in which the same record type plays the role of both the owner and member.

A network database consists of a collection of records connected to one another through links. A record is in many respects similar to an entity in the entity-relationship E-R model. Each record is a collection of fields, each of which contains only one data value. A link is an association between precisely two records. Thus, a link can be viewed as restricted form of relationship in the sense of the E-R model.

If the representation of a network database is viewed for two records, the relationship between them can be seen as one-to-many from both sides. As an example, suppose two record types linked with each other by an arbitrary links. Simulating such case using the relational database schema would yield in representing those two records with an address

field associated with each record type, then creating a third record type for representing the arbitrary links of the other two. Figure 3.14(a) shows the two record type representation. The relationship between the two fields of the middle record is a many-to-many. The outcome of the two record types is depicted on Figure 3.14(b)

Table A

Field	*Address*
A1	1
A2	2
A3	3
A4	4

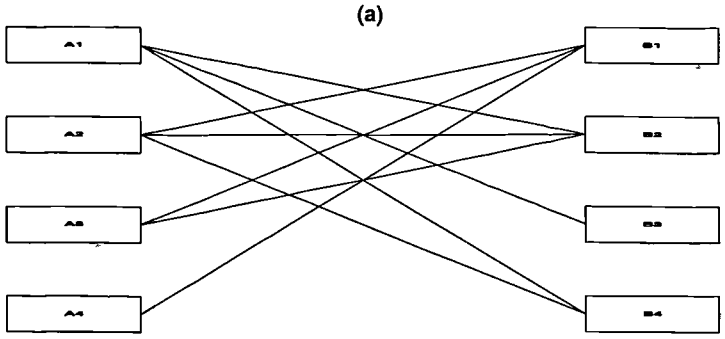
Links Between Table A and Table B

A Address	*B Address*
1	2
1	3
2	1
3	2
2	2
3	1
2	4
4	1
1	4

Table B

Field	*Address*
B1	1
B2	2
B3	3
B4	4

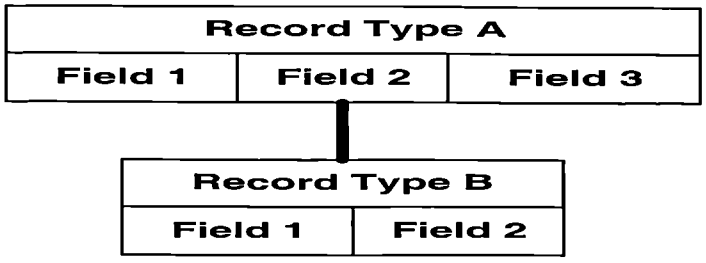
(a)



(b)

Figure 3.14

In the network database the links between the different record types is represented as an arbitrary graph. The relationship between any two record types may be either one-to many or many-to-many. So, the relationship between any two record types should be represented as a pointer between any two record types. This stipulation in the network database offers more freedom in the traversing process. Also, it minimizes the duplication in the records, which is considered as one of the drawbacks in the hierarchical databases. The network model differs from the relational model where data are represented by collections of records, and relationships among data are represented by static links. Figure 3.15 shows the code requirements if the network database for two record types simulated using the relational database engine.



Head (Record Type Name, Head Address)
Record Type A (Field 1, Field 2, Field 3, Next Record Type A)
Record Type B (Field 1, Field 2, Next Record Type B)
Link Table (Record Type A Address, Record Type B Address)

Figure 3.15

The network model is usually implemented by using pointers and circularly linked list. Most network DBMSs also present the option of implementing some set by clustering; that is, the owner record followed by the member records in physical contiguity for each set instance. The clustering of member records next to their owner record can be done only for a single set type that a record type participates in as member, because records can be physically clustered using the member based on only one logical 1:N relationship type. The set type that is used most frequently in accessing the records should be chosen for physical clustering. In many cases indexing or hashing on certain attributes of a record type can also be implemented on the ring file for fast access to individual record of a particular type.

3.4 The Object-Oriented Schema

Object-orientation provides facilities not existing in any of the other non object-oriented languages. The facilities for inheriting types and methods are very strong options offered only by object-orientation. Inheritance facilities, generalization or specialization means the same thing with different views. All those facilities are behind the flexibility gained by such technology when compared with the other non object-oriented technology.

Performance is a basic requirement in any database management system. In an object-oriented database system this is influenced by many factors, mainly stemming from the complexity of the object-oriented data such as the inheritance and the complex objects. Those, appropriate storage techniques for objects, as well as adequate indexing techniques, are needed to provide a good performance level. Complex objects so far proposed as the database schema for the OODBMS.

Polymorphism technique in the object orientation means the ability of different objects to respond each in its own way, to identical messages. It results from the fact that every class lives in its own name space. The names assigned within a class definition won't conflict with names assigned anywhere outside it. Both of the instance variables in an object's data structure and of the object's methods are protected. Method names are part of an object's interface. When a message is sent requesting an object to do something, the message names the method the object should perform. Because different objects can have different methods with the same name, the meaning of a message must be understood relative to the particular object that receives the message. The same message sent to two different objects could invoke two different methods. The main benefit of polymorphism is that it simplifies the programming interface. It permits conventions to be established that can be reused in class after class. Instead of inventing a new name for each new function you add to a program, the same names can be reused. The programming interface can be described as a set of abstract behaviors, quite apart from the classes that implement them.

The entity type schema in both the network and the hierarchical models is defined as a record type description. In the formal relational model it is defined as a relation schema, which is equivalent to a table description in the informal relational model. In the object-oriented model the entity type schema is defined as a class description. This leads to the fact that the content of the different schemas are defined different for each schema type where each may behave differently during the run time.

The entity set is known as the relation set in the formal relational model. In the network and the hierarchical models this is known as a record type instances. This is equivalent to collection of objects in the object-oriented model.

As a relationship between the similar schemas types the only models which got a real definition for the relationship in the building of the schema are the network and the hierarchical models. A static definition is created between the records at the time the different related record are created. On the other hand, there is no corresponding concept on the relational and the object-oriented model. The relationship in the relational model is established by using foreign keys while in the object-oriented model the relationship is established using references through object identifiers.

The object-oriented systems provide persistent storage for complex-structured objects. They typically employ indexing techniques to locate disk pages that store the object, which is the entity instance or row as defined in the informal relational model. Here the objects are often stored as byte strings and the object structure is reconstructed after copying the disk pages that contain the object into system buffers. A number of object-oriented and relational systems are using some form of client server architecture.

In order to increase the efficiency of the application in object-oriented databases, direct access to the attributes of objects is implemented as system-provided operations. This will also avoids the development of a large number of conventional methods by the programmer.

The object-oriented model has been criticized to be a step backward the navigation time compared with the hierarchical and network models. If the case is navigation using CAD or artificial intelligence applications, than the object model only provides the nested structure of the objects. Figures 3.16 and 3.17 show respectively the EER diagram of the conceptual schema for the library database and the O2 version of the class declaration of the EER diagram.

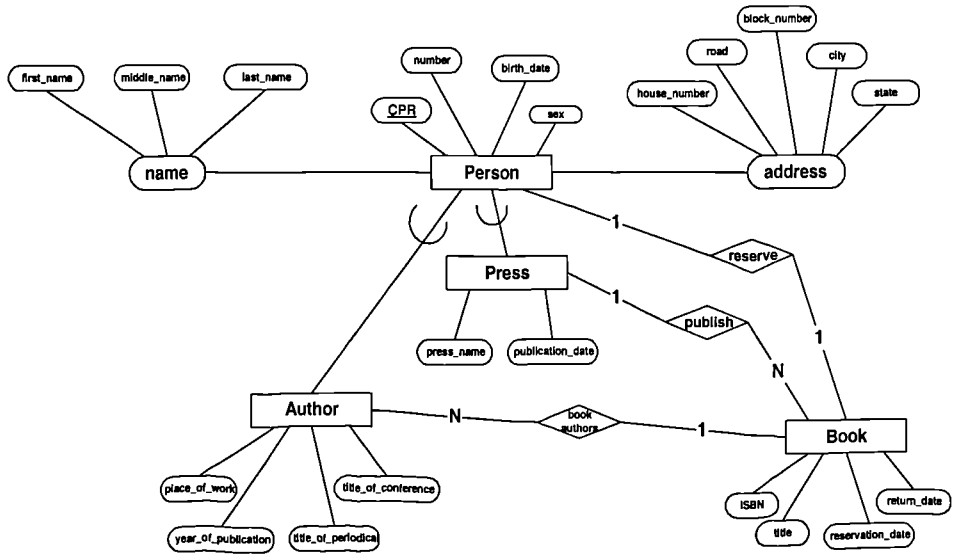


Figure 3.16

```
type Phone : tuple (        area_code : integer ,
                            number : integer ) ;

type Date : tuple (         year : integer ,
                            month : integer ,
                            day : integer ) ;

class Person
    type tuple (            CPR : string ,
                            name : tuple (      first_name : string ,
                                                middle_name : string ,
                                                last_name : string ) ,
                            number : integer ,
                            address tuple (     house_number : integer ,
                                                road : string ,
                                                block_number : integer ,
                                                city : string ,
                                                state : string ) ,
                            birth_date : Date ,
                            sex : character )
    method      method name : return value type
end

class Press inherit Person
    type tuple (            press_name : string ,
                            publication_date : Date )
    method      method name : return value type
end

class Authors inherit Person
    type tuple (            place_of_work : string ,
                            year_of_publication : Date ,
                            title_of_periodical : string ,
                            title_of_conference : string )
    method      method name : return value type
end

class Book
    type tuple (            ISBN : string ,
                            title : string ,
                            reservation_date : Date ,
                            return_date : Date ,
                            reserver : Person ,
                            publisher : Press ,
                            author : set ( Authors ) ,
    method      method name : return value type
end


method body method name : return value type in class class name
{
    body of the method
}
```

Figure 3.17

The object-oriented model is considered as the amalgamated model of the three other models. The outcomes of combining the relational model, with the hierarchical and the network model will result in a model similar as the object-oriented model. Of course, there are many additional facilities bounded with the object-oriented model known as object orientation capabilities. From those capabilities are the inheritance, aggregation, generalization, encapsulation and many other added facilities in which it will make the application easily maintained and new elements easily added to it.

For the purpose of storing data in an object-oriented database there are two proposed approaches. Those are known as the direct model and the normalized model. In the direct model, which is the one similar to the storage technique used in the relational database, objects are stored in the same way in which they are defined in the conceptual schema, that

is, the unit of storage is the same as the semantic unit. This means that objects which are belong to the same class are stored in the same file and each record in that file is an object instance of the class. An advantage of this approach is that transferring of the whole object is a very efficient process; sense join operations are not required to reconstruct objects that have been previously decomposed. On the other hand, accessing certain set of attributes within the class is considered as a disadvantage.

In the normalized model, objects are decomposed into atomic components where each is stored in different file. The relation between the different components is maintained by means of object identifications or OIDs.

Generally, an intermediate approach between the two proposed approaches can be adopted in which complex objects are decomposed where components are grouped together according to access patterns that are accessed continuously are stored in the same file.

The Object Database Modeling Group ODMG is the industrial party working towards standardizing the object based components industries. They have already achieved remarkable arrangements in areas like how object interfaces are defined, how operations are called on objects, how objects are distributed such as in the CORBA standard and what type of object services are used. An early agreement on an object database standard would undoubtedly eliminate most exploration of new ideas by DBMS vendors. In this case unless vendors extensively compared and evaluate each other inventions in the area consumers would not be able to get the best object-oriented databases technology features.

A very tangible feature of object orientation remains in its ability to have a collection of objects as a data type feature. The ODMG has defined four collection types to be supported by the standard. Those are the set, bag, list and array. Sets are an unordered collection of elements with no duplicates allowed. On the other hand, bags are the same as sets with possible duplicates. Lists are considered as an ordered collection of elements. Arrays, on the other hand, are considered as collection with a fixed number of elements that can be located by position.

One of the important strengths of the object-oriented model remains in its ability to sudden changes in the relationship design between its various classes. This ability has minimized the overall required changes in cases such changes happen to the business requirements.

Also, does polymorphism appear in the heterogeneous information sources interoperability? If yes, where? It seems that all the provided explanations and a different discussion happens in this are towards the programming part of the whole story of this broad area. The discussion was mainly towards simplifying the application programmers' mission in cases when modification and redesign of the actual life application is compulsory. In the IE polymorphism is applied by the nature of the prototype design. In this sense the user will be able to send the same query structure to different schemas without having to know the actual database structure he queries.

Questions remain to be answered; will the object-oriented schema be able to accommodate all the other models as partners in its complex data types? If yes, what are the likely impacts in the response time?

3.5 The Object Relational Technique

The object relational database management system is considered as the newest commercial breed that uses object orientation technique and at the same time using the relational schema as the backend data store. Also, object relational will support some of the object extensions needed by today's more complex applications.

Object relational database management systems add new object storage capabilities to the relational systems at the core of modern information systems. These new facilities integrate management of traditional fielded data, complex objects such as time-series and geo-spatial data and diverse binary media such as audio, video, images, and applets. By encapsulating

methods with data structures, an object relational database management system server can execute complex analytical and data manipulation operations to search and transform multimedia and other complex objects [Reinwald96].

The tenets in [Stonebraker90] have stated that the next generation database management systems are assumed to support richer object structure and rules, subsume second generation database management systems and has to be open to other subsystems. Also, the accompanied 13 propositions have also discussed the desired facilities to be in the new generation database management systems. Most of the work in this research has been concentrated in fulfilling tenets 2 and 3. Also, a consideration to proposition (1.3), (1.4), (2.3), (2.4) and (3.1) respectively.

As an evolutionary technology, the object relational approach has inherited the robust transactional and performance management features of its relational ancestor and the flexibility of its object-oriented cousin. Database designers can work with familiar tabular structures and the well-known data definition languages while assimilating new object-management possibilities. Query and procedural languages and call interfaces in object relational database management systems are familiar: SQL3, vendor procedural languages, and ODBC, JDBC, and proprietary call interfaces are all extensions of relational database management system languages and interfaces. And the leading vendors are, of course, quite well known: IBM, Informix, and Oracle.

In addition, extended relational and object relational are synonyms for database management products that try to unify aspects of both the relational and object databases. Although, there is not yet an official definition of what an object relational database management system is. Won Kim, founder of UniSQL, recently published a white paper describing a framework with which to evaluate the completeness of a product's compliance with seven major categories of capabilities of object relational databases. Also, efforts to interoperate between the object based and relational based schemas are considered as an ongoing research which this research tackles only part of the full picture.

3.6 Perspective in the Amalgamated Database Schema

When comparing the basic hierarchical database that is represented as a pure tree structure to the basic network database, many commonalties found with differences in the relationship between the different records. In the basic hierarchical databases the relationship between the different records always either one-to-many or many-to-one and never both sides relationship comes together. In the network databases the relationship between the different records is arbitrary. As an example, suppose there are two types records A and B. The relationship from A to B is one-to-many and from B to A is also one-to-many. There are many complicated scenarios that can be represented in both the hierarchical and the network databases may not be part of the experiment in this research.

The entrance for the hierarchical databases is limited and can be started through the hierarchies occurs in each hierarchical structure. In the network databases this can be done using the same philosophy by entering from number of record types, which is again limited. From the relational databases point of view this process can be applied from any of the relationship tables, which is unlimited such as the two previous schema types.

The simulation of both the network and the hierarchical databases using the relational engine, this is done without using any of the facilities offered by the relational database engine. This means that all the links between the different data records should be done manually to reflect the behavior of such legacy databases. This will involve designing two functions to enforce the behavior of both the one-to-many and the many-to-many relationships.

Network and hierarchical schemas use static or physical references, while the relational model uses logical or symbolic references. The network model can resemble the relational model if duplication for the key in the owner record is done in the member records. The hierarchical model also represents relationship explicitly, but it has series limitations compared to the network model. A record type in the network model can be a member in any

number of set types, while in the hierarchical model it can have only one real parent and one virtual parent. Generally, the hierarchical model is considered as inferior to both the relational and the network models in its modeling capabilities.

The network and hierarchical models resemble the object-oriented model in at least two things. First, both support some form of data nesting or cyclic objects. Both accept objects, which refer to other objects such as values of their attributes. The fundamental difference in this is that the implementation of cyclic objects in network databases tuff and requires artificial structures to be introduced in the schema. Second is, references in both are the same. This technique is referred to as object identifier, which is equivalent to pointers in the network and hierarchical models. Object identifiers are defined as logical pointers in object models, while these are defined as physical pointers in the network and hierarchical models and cannot, therefore, be used for checking referential integrity.

The point is not yet clear about object-oriented database technology i.e. its ability to mathematically prove its various concepts as in the other database schemas. It is well known that this kind of management systems could not directly access the various objects i.e. records directly without doing it through the class accompanied methods if available. Furthermore, the access to the various objects should be only applied according to what is in the accompanied class a method without even dropping single attribute.

Meanwhile, many projects are been undertaken for developing extensible database management systems. When these types are compared with the object-oriented database it shows that the extensible databases can provide physical or architectural extensibility. The extensibility in any system prepared by using any of the conventional languages is the system architecture dependent, which is not the same case if object-oriented technique is used. Object-orientation provides logical extensibility, which is the ability of defining new types of data and operation on an operational system.

When comparing the relational data model with the object-oriented model it is clear that the former does not directly support the complex object definition and it does not provide the notion of inheritance, whereas the latter support both. Also, the behavioral semantics of the objects in the former are dispersed in the application programs, while in the latter these are associated in the class definition. Additionally, the relational data model does not support the concept of object identity, which is the unique identification technique used by the object-oriented schemas for the purpose of making the object immutable to the outside world. Many comparison points have been made during the elicitation of data models earlier.

Indexes are basically considered as structure access techniques for the purpose to speed up the retrieval of records in response to certain conditions. They do not affect the physical placement of records on disk; rather they provide alternative search paths for locating the records efficiently based on the indexing fields. The most popular indexing techniques are based on ordered single-level indexes and tree data structures, which are the multiple indexes or B^+ trees. Indexes can also be constructed based on hushing or other data structures. The relational and the object-oriented databases only support dynamic indexing. This makes them flexible in a way to change the indexing key according to the requirements. On the other hand, the hierarchical and network schemas data models supports statically indexes in the insertion mode only. This makes them not flexible when change is required.

The idea behind the de-normalization process that would be undertaken by the IE, which will be applied to the normalized databases, will takes us to another broader ideas for the further broad amalgamated database schema. Normally, the purpose of the normalization process is to first get rid of the redundant data, which is basically a preservation process to the storage usage. Secondly, is to I provide better chances to apply the security constraint to a subsets of the complete data. Thirdly, which is the most know advantage, is to allow fast and rapped retrieving of the information in cases such having huge number of record.

If the de-normalized information will fit in a database that is similar to the spreadsheet than a balanced database having all the information in a single record could be considered. Of course, single record containing all the information related to the record, which is considered

as an object, can be understood easily by everybody. This is because it does not yet carry the different thought of design flavors of the systems designers. Also, the aim is to get the spreadsheet fully normalized in terms of storage usage and to be exactly equivalent to the original database system design.

This will lead us to the idea of having an amalgamated spreadsheet and supporting database operating system and security manager taking care of all the database operations and issues. All the database facilities such as the necessary screen building tools and the security assignment system are all available at one time. The implementation of such idea will cutoff the necessary time for building the tables, the attributes and the entity relationship diagrams for the database application implementation phase.

In general, for all the schema types the structure of the record is first red from the database management system tables that are holding the different attributes definitions with the types. Those are stored as metadata definitions for the entire systems working under this database management system. Although, some small businesses database management systems does not have the metadata information in a common place. The IE is first considering only the inclusion of the metadata of the information sources that will be shared with the global users. Second it conveys only the related metadata information parts to only the related information consumers in a fully controlled atmosphere. Third, it uses a special browser to display the metadata to specifically the related information consumers and not as the current Internet browsers that display the same pages to any requester.

This study is towards allowing database management systems implementers to create object like records that can be stored to and retrieved from any data store in any format. It allows clients to use their own designed interfaces regardless whether the record is stored in file system, relational database, object-oriented databases, or other data stores.

This thesis describes work in progress towards creating the heterogeneous information sources interoperation. It describes the interface used for storing and restoring records from heterogeneous information sources distributed over the Internet in a dynamic manner. Also, it describes all the related tools necessary for simplifying, securing, and retaining autonomy of that interoperability mission.

Chapter 4

A Perspective on Database Interoperability

Database interoperability refers to the ability to share part of the data someone owns with other global users. At the same time the interoperation process should hide the differences in all the stages of transferring the required information from one users' place to the other.

The implementation of DBMSs interoperability may be identified in three dimensions: distribution, heterogeneity, and autonomy as shown in Figure 4.1. Distribution means the system involves data at more than one location. Distribution may involve different approaches ranging from multiple-client/single-server systems to those which support peer-to-peer communication. The second dimension is heterogeneity, which refers to the differences in the design of the various system components such as: computer architecture, operating systems, networking protocols, programming languages syntax, the various application semantics, and of course database schema representations. Autonomy over databases is the third dimension and can be divided into three categories or levels as the following [Ozsu93a, Ozsu91]:

1. Design autonomy: where the local database design is completely left to the local DBMS user.
2. Communication autonomy: which is the level of freedom given to local DBMS user/creator to define the sharable parts of the local databases.
3. Execution autonomy: which states that DBMS users can chose how to execute a transaction over the databases for which they have access in any.

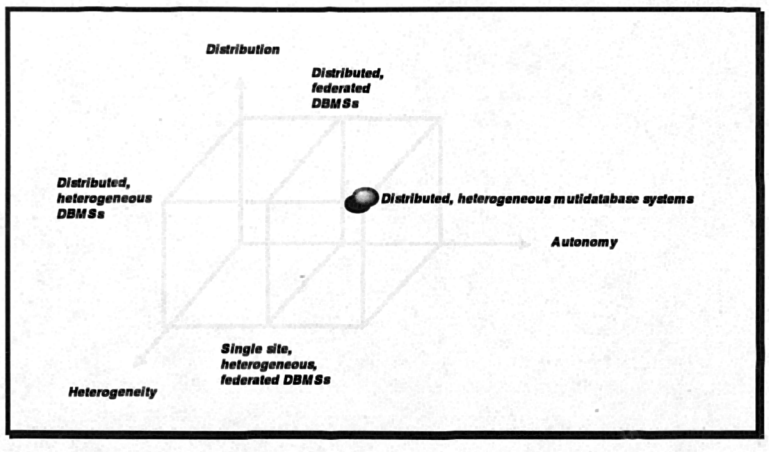


Figure 4.1

As shown in Figure 4.1, full heterogeneity is only considered if: databases in general is fully distributed, each of the data stores has full autonomy over its own site databases, and different types of database schemas can communicate to each other. Federated DBMSs are considered as semiautonomous systems, in that the component systems in a federated environment have significant autonomy in their execution, but their participation in a federation indicates that they are willing to co-operate with others in executing user requests that access multiple databases. It is clear from Figure 4.1 that the federated systems can either be distributed homogeneous or single site heterogeneous. Federated databases are not considered as the final solution to database interoperability. Although they support data

from many databases, as well as many DBMSs from many vendors, they are still considered as bridges built as an afterthought.

The federated approach to database management is very different from distributed database management. Whereas distributed databases are often carefully planned, federated databases are often evolve as a result of external circumstances. Unlike a true distributed database a federated database characterized by loosely coupled applications and diverse data models [Won90]. Although it has strong contribution in the interoperability of databases, federation is not yet considered as the final solution. The federated approach through import/export schemas makes integration more tractable in large multidatabase systems although this approach gets less tractable as the system size grows [Heimbigner85, Sheth90]. So building specific bridges between database islands is not a feasible process when it is intended for a long period and changes to the database internal schema representation are considered. For this purpose, database research should work towards building a solid infrastructure for the interoperability of the databases taking into consideration allowing different schema types data stores to interoperate with each other and by offering the highest security environment to all the interoperating data stores.

Management of heterogeneous databases, which is the second dimension in the taxonomy of Figure 4.1, is one of the most important areas of research and engineering today [Dayal84, Litwin88]. The ultimate objective of research and development in heterogeneous databases is to produce a system and a set of user aids, which will allow the users to access a set of heterogeneous databases, *using the local database language*, as if they where a single local database. Introducing heterogeneity may not involve significant additional difficulty to the final solution. This, of course, is true only from the perspective of database management. There may still be significant heterogeneity problems from the perspective of the operating system, the attached communication protocols, and the underlying hardware [Won90].

A legacy information system is any information system that significantly resists modification and evolution to meet new and constantly changing business requirements [Brodie95, Jeffery95]. Legacy information systems fall into three creation categories, which are:

(1) Data are mixed with the program code and both are mixed with the operating system which is the most difficult to deal with.
(2) Only data is mixed with the program code and both are separated from the operating system.
(3) Data is separated from both the deriving programming language and the operating system. Although this research aim is towards total information sources interoperability most of the design algorithms are based on this category.

Existing database schemas are not designed so that they can interoperate with other schema types, which make the problem of database interoperability not an easy problem to deal with. As part of the research in database interoperability, a study should be conducted on the existing DBMSs to determine their ability to interoperate with other similar or different types of databases. The solution would list a number of policies, algorithms, and other roles to be applied by each of the DBMSs in order to be able to interoperate with other distributed and possibly heterogeneous databases.

Building bridges between different types of databases is, to an extent, support for the heterogeneity interoperability problem but does not solve it. The full solution would only be considered if it is a generic solution in a sense to be transparent, as well as, dynamic. If any changes are made to the databases, these should directly be reflected in the local, as well as, the global world [Brodie95].

Database heterogeneity can fall into a number of different areas. The heterogeneity of databases are defined by [Bouguettaya95b] as follows :

(1) Transaction Management is a mean to maximize concurrency and ensure consistency by using transactions to organize and control database

activity are the hardest to interoperate [Sheth90]. This is especially true in distributed databases willing to interoperate. [Baker90] has addressed the reliability and execution of transactions in multidatabase systems.

(2) Integration of different schemas of the same or different models.
(3) The interoperation of different query languages.
(4) Interaction of different database applications.
(5) Other lower levels involving operating systems and communication protocols, which are considered outside the database research realm.

Because interoperability, in general and not specifically for databases, is not a simple task, there should exist a technique capable of handling all the issues needed in such environment. Object-oriented technology is a promising and powerful technology, which is considered as the technology that will cooperate in solving such problem. Object-orientation has got the properties of encapsulation of operations, distribution of autonomy, processing concurrency, and reactive control structures. It may be the most appropriate model for designing the multidatabase system, of course after resolving the problems of the networking heterogeneity [Browne93]. Because of these reasons, and many others, it has been suggested that object-oriented technology may solve the interoperability problems. It can contribute in handling the heterogeneity aspects of interoperability by encapsulating systems and hiding their heterogeneity. It may also contribute to the uniform handling of object distribution. However, the more difficult problem of autonomy are the problems created by the heterogeneity in the layers where those databases are working under them such as the operating systems and the communication protocols that still needs to be dealt with [Ozsu93a]. Furthermore, the three-tire client/server application programming techniques and open systems technology in conjunction with the object-orientation may have an excellent contribution to the interoperability process.

There is now a realization that DBMSs and operating systems perform both similar and complementary functions and that a better cooperation paradigm needs to be found. This topic has not yet been fully researched and understood, and still there is no clear understanding of how an operating system should be structured to provide adequate support for the DBMS functionality [Ozsu91].

Moreover, database security is an important issue in today's distributed systems. Many security services have been designed and implemented on many different platforms. However these implementations are often not compatible [Decker92]. The incompatibility of the security in the DBMS created the discrepancies in today's databases. A unified security manager to manage a single site running number of database platforms each running a different schema type would make the control of the security issue as easier task.

Gateways are those bridges built when there is a need for them to convert between different schemas. They are not considered as a dynamic solution engines for schema translation process. Examples of gateways are Ingres/Star and Sybase, which provide access to several standard relational database engines as well as more widely used for file systems such as RMS and VSAM. Gateways only offer an interim solution to heterogeneity. Also, the existing gateways do not incorporate the business rules of the information source.

According to Chris Date's rules [Date95], the distributed database is a collection of data distributed over different computers in a network, and that despite being physically dispersed, that data appears to programmers and users as a single database and this is sometimes called the single-image model. Full database interoperability can be only accomplished when and only when Date's Distributed Database rules of interoperability are fully met [Edelstein90].

Much research has been done for the purpose of providing some sort of compatibility between different databases. Those databases range from those providing only textual facilities to those providing full multimedia facilities such as picture, voice and video. However, the problems of interconnecting heterogeneous networks and databases remains to be solved before integration of diverse information sources can take place [Browne93].

Many articles and books in the literature have defined the overall problem of interoperability in general. None of the existing works until now is considered as the ideal solution to the problem. Of course the problem is so broad and need not only to deal with practical solution, but also need to standardize policies and procedures for this purpose. Interoperability does not only concern databases, but is it meant by all data types (i.e. databases, flat files, spread sheets, audio and video), including text based and non-text based data by which this research deals with the known database schemas (relational, hierarchical, network, and object-oriented). The total picture of this research is any manageable information source.

4.1 Requirements Analysis

By analyzing the traditional assumptions on data consistency and schema stability it is found that most of the existing solutions has been built around those two assumptions by basically having a global valid database schema that has the central authority of the database administrators. However, in the proposed solution a definition of a dynamic mechanism using the Internet as the backbone with keeping all the interoperating data sources know about each other and get the updates in a dynamic manner has been undertaken.

A real world of multidatabase system may include thousands of information sources managed individually by different database management systems by which the heterogeneity in this case is nature and unavoidable. The information sources maybe heterogeneous in the logical data structure, in type and attribute naming, and probably in representational semantics. The heterogeneity in the information sources has resulted in the difficult communication between the information sources. Some information sources have similar data contents, but different representations. This is strongly true for such data sources created and managed by separate organizations. Moreover, different heterogeneous factors my have different effects in the data sources interoperability such as the heterogeneity in the types of data model, query languages, and transaction management protocols.

Representing data with different data modeling tools creates lots of heterogeneity problems because of the inherited powers and limitations of individual data models. Heterogeneity in query languages not only involves the use of completely different data access techniques, but also covers differences in languages even when the individual systems use the same data model. Also, different query languages that use the same data model often select very different methods for expressing identical requests such as DB2 uses SQL, while INGRES uses QUEL.

Existing solutions prototypes of heterogeneous distributed database systems can be broadly divided, based on their philosophy of integration, into three categories: total integration, partial integration and interoperability. Total integration is when storing all the database components under a centralized database management system to provide a unified global view of all the data to the user who can pose the query on this global schema. The other extreme, which is the interoperability, does not provide a global schema. Rather, the user is presented with functions in visibly distinct schemata, which bear resemblance to schemata in existing databases. The intermediate category of partial integration is considered as a hybrid of total integration and interoperability. Systems such as MULTIBASE [Smith81], MERMAID [Templeton87], NDMS [Staniszkis], IMDAS [Krishnamurthy87] and ADDS by [Breitbart86] are in the category of total integration. MRDSM [Litwin86] is in the category of interoperability and PRECI [Deen85] is in the category of total integration. The focus of this thesis is on total integration of partial information sources selected dynamically by both information producers and information consumers.

4.2 Autonomy Requirements in the Interoperation

Autonomy usually refers to the distribution of control, not the data. It indicates the degree to which individual database management systems can operate independently. Autonomy is a function of number of factors such as whether the component systems exchange information, whether they can independently execute transactions, and whether one is allowed to modify them [Gligor84, Gligor86]. The considerations for this research is to design the information sources autonomy in a way to provide a repartitioning on the interoperating information

sources without violating any of the DBMS primitive facilities. This should provide the information source owner the capabilities to define logical views on his information sources and define the access rights on the information source up to the level of permitted users. Requirements of an autonomous system have been specified in a variety of ways. For example, some researchers says that the local operation of the individual DBMSs are not suppose to be affected by their participation in the multidatabase system. Also, they stated that global queries on certain information source should not affect the manner in which the individual information sources process local queries and optimize them. Additionally, compromisation for the system consistency or operation should happen when new information sources join and leave the interoperation.

On the other hand, some researchers have segregated the autonomy into three dimensions. The design autonomy refers to the freedom that each information source could use in terms of the data models and transaction management techniques. The communication autonomy refers to the freedom each individual information source should have in terms of to what type of information it wants to provide to the other information sources or to the software that controls their global execution. The last type is the execution autonomy that refers to the way each information sources in the interoperation want to execute the incoming transactions. The execution autonomy requires that the submitted transactions should get translated to the way it can be understood. The proposed solution has considered all the above autonomy proposals.

4.3 The Cooperative Interoperation in the IE

The cooperative interoperation could be best described by the mutually benefit relationship between number of information sources in terms of exchanging information and using each other information results in a cooperative manner. To expand the information samples it is most desirable to get similar information from others and set a cooperative share with them. The only obstacles behind this scenario if the others information sources either using different schema type and/or is fully designed different than my local information source. The IE has considered the cooperative interoperation in a way by adding additional layer that will deal with all the conflicts that may exist during the interoperation process. Part of its tasks is to manage and coordinate the syntactic and semantics discrepancies. Such process would be accomplished through using a management interface with the IE.

Chapter 5

The Possible Cooperation Between the Different Schemas

Database schemas are the main buildings of any database application. The different existing schemas (relational, hierarchical, network and object-oriented) may cooperate with each other where the cooperation mechanism is governed by each schema constraints. On the other hand, the amalgamated database is the technique using more than one type schema at the same time. This seems to be the future of the database management systems remains to be achieved. Either new DBMS will offer connectivity with different types of schemas, or they will offer different schemas under a unified platform. This consideration is true because of many reasons where one of them is hierarchical systems are much stronger than the relational in the retrieving and the reporting capabilities. Of course, each of the schema representations has their own advantages and disadvantages. Depending on the nature of the application requirements the amalgamated database can cooperate in developing the most feasible, interoperable and tractable database management system. This chapter is mainly examining the different possible cooperation between the different schemas and the different constraints belong to each of the schemas that influence the cooperation. The main contribution of this chapter is the highlight of the possible cooperation between the different schemas in the light of the individual schema different constraints.

One of the research targets in distributed heterogeneous database world is to achieve an amalgamated schema support systems by which no lost, no rewrite, no migration and no even amendments required to be applied on the legacy database systems. But still there are many things govern this technology by which the interoperation techniques must respect to get things done correct. Assume someone is setting in one side of a local area network with a LAN based DBMS such as the relational DBMS and wants to access a remote mainframe hierarchical DBMS. He/she therefore will be able to set a relationship between my local relational databases and the remote hierarchical databases. Many operations to be assigned such as add, delete, edit, query and any other operation that can be applied locally it should be true on the remote databases.

Schema type is normally only judge about the constraints, which forms the overall behavior of the database system. Even the newly used DBMS schema, which is the object-oriented, have the difference from the other schemas in having data types complex type of type schema or record in ordinary database language. The rest of this chapter is a discussion of the possible cooperation on the four type schemas (relational, hierarchical, network and object-oriented). With an example the possible cooperation in the four schemas will be explained.

By the known terminology the database is the store, which may consist of one or many tables. The most important data element that users are normally dealing with is the table. The IE, which stands for the Interoperation Engine, is designed to deal with the database table level [Ashir2000]. Local databases may cooperate with many remote tables they may spread over in many databases. The IE is assumed either to read the metadata from an already defined database or by using the IE GUI it will be possible to create the text that will create the database definition.

The cooperation between the distributed data sources is normally done on the table level where the metadata of the data sources are known for both sides. The IE considers the main data source is the one where the local application program will be written against it, which

mostly is a local table. The IE design is done so that all the cooperating data sources to be accessed by a local application are remote. In this case, the metadata of the accessed data source will be mainly the variables defined by the local application, which the local IE will take care of defining and creating the data sources that will be accessed. Assume dealing here with three remote data sources (i.e. tables) such as T, which is the local data source and T_1, and T_2 as remote data sources. The operation between them is T Union T_1 Intersection T_2. The IE will first apply T Union T_1, then the new T, which already contains the result of the first operation with T_2 as Intersection operation between them.

In this chapter all the different cooperation cases that could happen between the heterogeneous and possibly distributed data sources will be considered. All of the different integration cases between the heterogeneous information sources are considered to be supported by the IE prototype such as unification, intersection and differences in the records of the cooperating information sources.

5.1 Representations of data models

The representation of relationships in data models differs in the different data models. In the relational data model the relationship is represented as foreign key attributes in one relation that reference the primary key of another relation. The tuples in both relations that have matching values in the foreign and primary key attributes are logically related and not physically connected. On the other hand, similar one-to-many connections between two record types in the network model are explicitly represented by the set type construct and physically connected by the DBMS. So, relational model uses logical references while the network model used physical references.

Compared with the network model, the hierarchical model has serious limitations. Record type in a network model can be a member of any number of set types while in the hierarchical model it can have only one real parent and one virtual parent. Hierarchical models mainly have serious problem in expressing many-to-many relationship types and usually solutions to such problems are difficult. In general, the hierarchical model is considered as an inferior to both the relational and the network data models.

Object-oriented models uses the object identifiers (OID) for the relationships that are almost similar to the foreign key in the relational model, except that internal system identifiers are used rather than user defined attributes. The object-oriented models support complex object structure by using tuple, set, list and other constructors. Also, the object-oriented models support many techniques such as inheritance and the specification of methods. The object-oriented models resemble to an extent the nested relational model, which supports the creation of hierarchically structured relations in addition to flat relations.

5.2 Integrity constraints in data modeling

Constraints are the different stipulations assigned by the DBMS to insure the integrity of the data in the process of inserting new information and destroying certain information from the database. Throughout the DBMS lifetime there are number of constraints assigned while information added, amended and deleted from the database. They are domain constraints; key and relationship constraints; general semantic integrity constrains; inherent; implicit and explicit constrains.

Domain constraints are those ones related to the different data types assigned for the attributes in the tuples. This includes voice, video and the other complex data type supported by the object-oriented based data types.

Key and relationship constraints are basically the primary unique keys assigned mainly by the relational, network and the object-oriented model. The key is known as the unique field and the object identifier respectively in the network and the object-oriented models.

General semantic integrity constraints are those which cannot be specified on the bases of key and relationship constraints. The mechanism by which defining such constrains is still

refined in commercial database systems. In general, there are two proposed approaches namely, the procedural and the declarative. The former is mainly the constraints are coded into appropriate transactions by the programmer. The former approach uses special constraint specification language so that if changes to be made in the constraint this will not involve recompiling the program again. In other words constraints enforced while they are not part of the original application. Research on making this approach more efficient continues in areas such as rule systems and deductive databases [Ozsu91].

Inherent constraints are those set of built-in constraints associated with the constructs of the data model. These kinds of constraints need not to be specified in the data definition language of the schema creation process. Rather they are inherited properties of the data model construct such as the relationship in the hierarchical data model is always one to many between the tree levels and in one direction only.

Implicit constraint are those enforced by the DBMS data definition language during the schema definition such as the unique key, which is recorded in the DBMS catalog and enforced automatically by the DBMS software. On the other hand, the explicit constraints are the general semantic integrity constrains.

The IE is designed to consider the different constraint assigned by the information source and its running application. This is to retain the integrity of the cooperating information source. Although assigning new constraint and modifying existing once will involve changing for both the IE knowledge and the application side this will retain the accuracy of the data.

5.3 Schemas cooperation process in the IE

Part of the IE responsibilities are the definition of the cooperation operations between the distributed heterogeneous tables and the gluing operation between the cooperating information sources attributes that are equivalent. Figure 5.1 explains a partial definition applied by the IE for the cooperation process between the schemas in relational terms. As shown in the figure, the subsystem may consist of one or many schemas, and the schema may consist of one or many attributes.

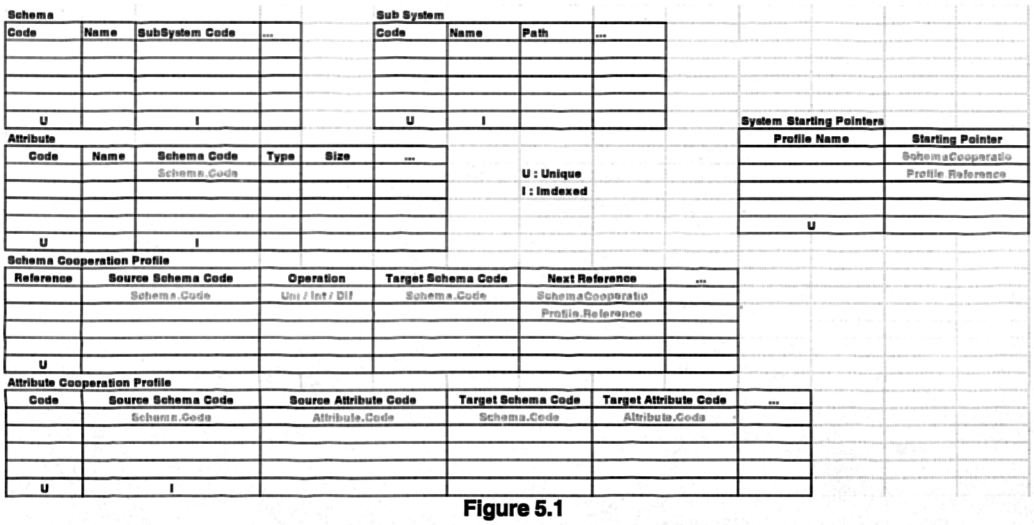


Figure 5.1

The operations between the different tables are taken from the schema cooperation profile table shown in Figure 5.1. The attributes by which the operation is applied on them are taken from the attributes cooperation profile table of the same figure. Here the IE takes the responsibility of converting the schemas from one schema type to the other. Appendix E provides a complete schema cooperation example based on the above scenario.

The table profile is only the real definition of the table that will be accessed by the application programme. The actual generation of the records inside the table will only be done when

requested by the application program that using the table profile. The generation of the table profile will follow the algorithm given in Figure 5.2.

```
1. Notes:
2. The gray means note.
3. The boldface means database information as shown in figure 4.
4. Italics means fields as shown in figure 4.

5. The algorithm:
6. Open the schema cooperation profile table
7. Take the link for the table by keeping the source schema code

8. Open the attribute cooperation profile table
9. Get all the table profile fields, their types, and size

10. Create table profile <Name> as
11.     Field 1   :   type,    size,
12.     Field 2   :   type,    size,
13.     Field 3   :   type,    size,
14.     .
15.     .
16.     Field N   :   type,    size,
17.     Path of the profile,  Access profile
18. Close

19. If profile depend on source table then
20.   Append all source records into the table profile
21. Else
22.   This means the source table is not exist and will be empty table
23.   Append all first cooperating data source related records in the table profile
24. End

25. Parallel process
26. Open the schema cooperation profile table

27. Go to the reference of the first cooperating table in the schema cooperation profile table

28. Memorize the operation

29. Open the attribute cooperation profile table
30. Do while the schema cooperation profile next reference not = 0
31.   Set the next reference
32.   Do while the attributes cooperation profile source schema code =
33.         schema cooperation profile source schema code and
34.         attributes cooperation profile target schema code =
35.         schema cooperation profile target schema code
36.         Other checking related to security done at this point

37.         The local IE to the target schema will extract the real data from the data store
38.         Store all the attribute values into temporary memory variables
39.         Check where to create the temporaty table profile based on the definition line 10-18
40.         Create a temporary table profile
41.         Append the record contents into the site temporary table profile
42.   End
43.   Go to next reference in schema cooperation profile
44. End
45. Close the attribute cooperation profile table
46. Close the schema cooperation profile table
```

Figure 5.2

As a further step for adding the related records to the local table profile, which will be accessed by the users' local application, the algorithm displayed in Figure 5.3 is responsible for carrying out this process when it is required. The accumulation of the remote data to the local table, which is called a local profile, will be done in series. This is because it is not possible at this stage to do the process in parallel as been done while preparing the remote tables to be readable by the local application schema. As an example, the unification, intersection, and difference operations requires the table to be completely known prior to applying the operation.

```
1. At this stage all the cooperating databases are ready for applying the operations in the schema cooperation
2. profile table. The main profile will be the source schema, which the application deals with. all the updates
3. from the target schemas will be done to the source schema. if necessary target schemas will get converted
4. to behave as the source schema.

5. Open the schema cooperation profile table

6. Go to the reference of the first cooperating table in the schema cooperation profile table
7. Memorize the operation

8. Do while the schema cooperation profile next reference not = 0
9.     Set the next reference
10.    apply the operation between the source schema code and the target schema code schemas
11.    Append the record contents into the source table profile
12.    Go to next reference in schema cooperation profile
13. End
14. Close the schema cooperation profile table
```

Figure 5.3

Line 8 of the above algorithm shows the tracking of the cooperation process is done according to the pointer defined by the next reference. This operation is done after the preparation of the distributed heterogeneous schemas homogenization is completely done by the local IEs and the cooperating tables are ready and equivalent to the local table which is considered as the main that the local application will be written across.

5.4 The Heterogeneous Schemas cooperation

As a very popular standard, the relational algebra operations are the standard mathematical operations on sets. They are applied on the relational model, which is defined to be a set of tuples and can be used to process the tuples in two relations as set. Several set theoretic operations are used to merge the elements of two sets in various ways, including UNION, INTERSECTION, and DEFFERENCE. These operations are binary as applied to two sets. The two sets should have the same number of attributes and that each pair of attributes has the same domain. The proposed distributed IE layer is designed to take care of the compatibility of the different cooperating information sources. It will play the rule of the homogenization layer between the incompatible unbalanced information sources.

The cooperation processes in the IE are considered to be the three relational algebra operations. They are the ADD, DELETE, and UPDATE, which are normally applied by the operational algebra operations. Also, the relationship between local and global information sources could also be applied through the IE.

The cooperation between the different distributed schema types may occur in many shapes. It could be a combination of multiple distributed tables according to any of the said cooperation process. This could also be further combined with the normal database system operations. So, the cooperation process of local information source with global information sources via the IE will take over the responsibility of ensuring the appliance of the various constraints belongs to the local and global information sources.

In this section the case of making each schema type cooperate with the others is illustrated by examples. The various rules each IE plays in facilitating the process of data interoperability are also discussed. The IE in each of the sites is responsible to make the attributes of the different sites compatible with each other and make sure the cooperation between the tables is a successful process.

The relational engine, which is the first schema example, provides dynamic pointers between the attributes and the tuples in the relation. Dynamic pointers are established in the time of binding between the relations. Dynamic binding is considered as a drawback when the process speed was low, but with the improvement in the process speed this drawback has become negligible. Of course, the statically defined pointers are considered as an advantage over the dynamically defined pointers, especially in retrieving operations. This section

negotiates the process of cooperating a relational schema as the source schema type with the other types. The negotiation is supported by examples as possible to clarify the idea of interoperating heterogeneous and possibly distributed database schemas.

Interoperation of relational data sources with another relational one is considered to be the normal case. If the source file is equivalent to the target then the possible connectivity between both could be the union, intersection and difference operations. In the case of non-equivalence than the possibility is relationship bearing in mind that both have connection foreign keys. The different interoperation operations will depend on indexes that both sides provide rather than migrating to unified pool as conventionally done. Regardless of the number of the cooperating data sources the IE will handle the different operations in parallel. Only the migration to a unified single pool by the IE will be applied in the case of reporting from all the cooperating data sources and will be done temporarily until the request is over. Further, the IE will take care of the homogenization process in the unbalanced information sources in case unification, intersection and difference operations to be applied.

If the target data source is hierarchical then this will be seen by the local relational database as a single record. The local IE of the target hierarchical database would convert these hierarchical schemas into a single table to make it simple and straightforward when accessed or queried by the source application. This conversion will only be done if the source table is to merge with the target by either unification or an intersection operation. If the operation between the two tables is a relationship then a foreign key in the relational database linked with a key in the stub record of the hierarchical schema side should exist. The IE in the hierarchical database side will take care of searching the required records and converting them to be readable by the source system. In this case, the IE will be leasing with both information sources as a custodian. It will send the pull procedures to the target information source, which is the hierarchical, and will send the result to the source IE prior to pulling the information by that source database.

There are many other scenarios where some are complicated. For example, if the target data source, which is the hierarchical, permits addition of new records then the source data store, which is the relational, has to apply the target data source constraints. The constraints are either to be enforced by the source database or the target database that the constraints are originally belongs to. In this case, the mediator which is responsible for the interoperation process is suppose to be responsible for enforcing the necessary constraints in case of adding new record and alteration of an existing records. It is considered a complicated case if the added new records are to be done in the source relational database while the data integrity constraints are done according to the target data source, which is the hierarchical. In this case the relational data source design has to be in a position able to cater and accept records in the light of their local constraints stipulations. Different constraints will be recorded by the related IEs to keep records flow between different information sources balanced as will be shown later. As an example for the above discussion consider the following equivalent relational and hierarchical schemas which are presented in Figure 5.4.

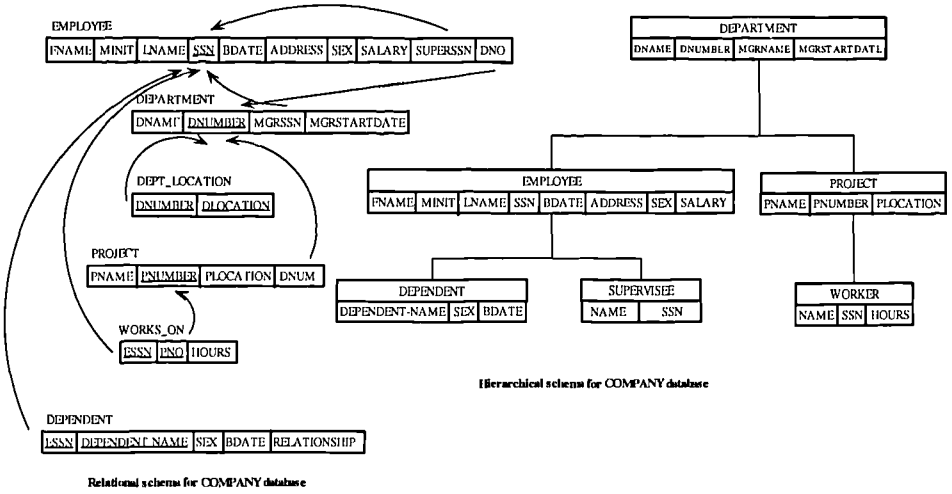


Figure 5.4

The main restriction in the hierarchical schema is that for reading or writing operations the start has to be from level 0, which is the department record. The pointer then goes to the first record of level 1 records and continues until reaching the last level record. In the last level record the pointer follow the next point until reach the null pointer it then goes one level up and again take the pointer to one level down until it traverse all the records. Figure 5.5 the illustrates the movement of the pointers in the hierarchical schema.

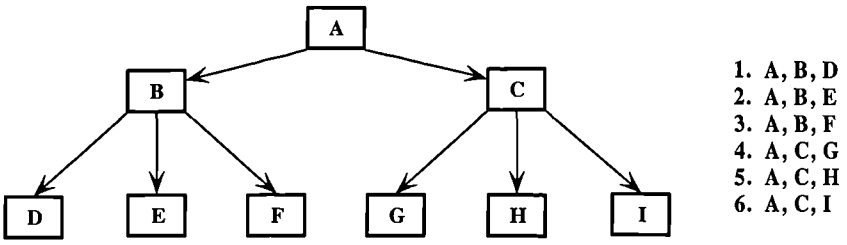


Figure 5.5

The first assumption is to unify the local relational schema, which is the employee table, with the remote hierarchical schema employee record. In this case the relational schema is the local schema and the hierarchical schema is the remote schema. The reading in the remote hierarchical company will start from department as level 0, then employee as level 1, and finally with supervisee as level 2. The relational employee table is equivalent to data from three hierarchical records. Those are the department, employee and supervisee. Collecting attributes belong to a single record type from a hierarchical schema is always involving traverse the entire tree to fetch the records. This may be problematic in some very large hierarchical systems.

The relational database, which is our source database, is to have a unification operation with the target database, which is the hierarchical COMPANY in the above example, it will involve migrating temporarily all the hierarchical records to the relational shape before applying the unification process between both of them. The migration here is done only when required by the source application in cases such as reporting involve all the records in the target database. In cases such as querying which only require a single record of specific number of records this will involve only migrating a single record at a time. The local IE will take the responsibility of the full and partial migration of the target schema according to the source application requirements. This mechanism is made according to the assumption that the differences in the operational and the behavioral characteristics of the storage compared with the memory are not distinguishable.

The cooperation between a source relational and a target network schema is a simple case because the network model is the closest model to the relational. Migrating network schemas to equivalent relational schemas is a simple case since the later is the closest in shape and

behavior to the former with one difference related to the relationship between the different records. As explained earlier the network model differs from the relational in the relationship references. The references are defined statically in the network model and dynamically in the relational model. The local IE is responsible for the negotiation between the different schemas in a compatible atmosphere.

Object-oriented databases are also not differing much from the relational databases in terms of schema representation and this make them easily converted to relational schema. Class of objects is equivalent to table in relational terms. Each object is equivalent to a record or row in relational schema. Because object-oriented databases support complex data types there is no need for the relationship operation to be there sense it is supported by default when using complex data type facility. Figure 5.6 shows the equivalent O2 object-oriented schema of the relational COMPANY schema. The only difference between both is the references used by the object-oriented model as a replacement for the relationship facilities in the relational model. The most important benefit of such facility appears in writing the program code. Object-orientation techniques facilitate writing and modifying the programming code.

```
type Date: tuple ( year :   integer,
                   month :  integer,
                   day :    integer    ) ;

class Employee
   type tuple ( first-name :              string,
                middle-name :             string,
                last-name :               string,
                social-serial-number :    string,
                birth-date :              Date,
                address :                 string,
                sex :                     character,
                salary :                  string,
                supervisor :              Employee,
                department :              Department    )

class Department
   type tuple ( name :              string,
                number :            integer,
                dept-manager :      tuple ( manager :   Employee,
                                            start-date : Date ) ,
                locations :         set ( string ) ,
                employees :         set ( Employee ) ,
                projects :          set ( Project )  )

class Project
   type tuple ( number :                    integer,
                name :                      string,
                location :                  string,
                responsible-department :    Department,
                works-on-the-project :      set (   worker :        Employee,
                                                    working-hours : integer )    )

class Dependent
   type tuple ( employee :      Employee,
                name :          string,
                sex :           character,
                birth-date :    Date,
                relationship :  string     )
```

Figure 5.6

If the assumption that the source application, which is originally written against a relational schema, queries about only certain cases from the cooperating data sources then only a search for this type of data will be done in parallel against all the cooperating data sources. The final view of the table profile will be according to the conditions specified in the first table given in Figure 5.4..

The second case e is when a source hierarchical database and the targets are from heterogeneous schemas. The first case is when the source is hierarchical and the target is a relational database. For the relational database the IE follows the relationship between the different tables and extracts the content of the record consisting of all the information from all the related tables into one single record. Only when doing so the migration from the relational back to a hierarchical schema could be achieved. Also, for the network and object-oriented schemas the process of migrating records from them to an equivalent hierarchical schema will involve first retrieving the record as a single record contains all the equivalent attributes to the source hierarchical. As a subsequent step the single record is to be read and added as a new

record in the hierarchical schema in the case the source schema is ready to register the distributed migrated records temporarily. This procedure will update the hierarchical database from global heterogeneous databases. The added records could be deleted after finishing the query if records being stored in the original database. Otherwise, according to the IE recommendations, the new records will be added on a memory space by which it could be flashed anytime.

As in the second example, the operation involves unifying the hierarchical database contents first with the source relational database then intersecting the contents with the object-oriented database records or the objects as called by this schema type. In this case either the migration of the hierarchical COMPANY will be done in the target premises and will be read by the source IE as a ready converted schema or will be read and directly stored as relations in the source premises, which is a general case managed by the IEs in all the sites. The local IE based on the source application requirement takes the decision of whether to migrate part or the entire target schema. In the case if the source application only requires reading only few records based on certain condition than the IE will only ask for that part of data to be migrated, while having full picture for different users.

The network schema has similar case as the hierarchical schema in case it is the source schema when taking records from different distributed schemas. The migration will be done only after de-normalizing the target information sources data to a single flat record consisting of the entire database attributes required by the inviting schema. The IE would take care so that it is done in a way which does not break the business rules or constraints that has been assigned initially by the DBMS and the application managing that information repository. The de-normalized information will be sent a record at a time by the individual cooperating IE to the calling IE either in parallel or serially as one repository at a time in the case of the operation is intersection or difference.

The IE will keep returning the related tables to a single non-normalized one for the sake of keeping the cooperation process as simple and reliable as possible and to make the pulled information available to the information consumer as long as he/she needs them. The de-normalization process can ensure the simplicity of retrieving any information from any single non-normalized table in an easy manner. Also, it will facilitate the migration process between the different schemas. Furthermore, the de-normalization can act as a standard for the migration process between the heterogeneous information sources through a middle engine as the proposed IE prototype.

In contrast to all other schema types the object-oriented schema can make the maximum use of the facilities provided by the IE, because the IE prototype mainly built using the object-oriented technology. No doubt that object-orientation has changed the way people thinking about data in a very simple and straightforward manner. Local IEs are considered as temporary pools for the exchange of information among the heterogeneous data sources. By using the techniques provided by the object-oriented databases, which are mainly running object-oriented programming languages, it will be an easy task to get the necessary data from any schema type that is built around object-orientation.

5.5 Wrapper usage in the IE

Wrapping is the technique of encapsulating part of the programming language in an envelope and executes the contents of this envelope when required by either the local or a remote open system capable operating system. The main purpose behind the wrapping technique is to integrate the legacy applications into an object-oriented architecture, and perform this integration in an inexpensive and timely manner. A wrapper usually communicates to the legacy systems using its native communication facilities. The resulting integration is assumed to be functional and robust in a sense that not much change is required for getting the integration of the legacy database data.

The cooperation between the distributed and at the same time heterogeneous databases is mainly done either querying certain conditions or reporting with no conditions which both doesn't required code wrapping facilities. The only requirement for wrapping technique to take

place is in case an alteration to the original data source requires updates to be undertaken by global users who not talk the same language as the information source owner. Chapter 6 is a thorough explanation of the application design requirements in case interoperation is required.

The interoperation engine IE framework has been built around the consideration of having most of the business and database roles defined in the engine interface to minimize the usage of the wrapping. Minimizing wrapping also adds strength to the solution in terms of the generality and maintainability.

Also, part of the IE implementation is to buildup a framework to be undertaken by the information sources owners willing to share and cooperate between themselves in a dynamic and systematic manner using the Internet. This objective is mainly aiming to initiate a design implementation framework and standards responsible to facilitate the interoperation mechanism for the distributed heterogeneous information sources willing to cooperate. The scope of the research is eliminated only on the design of the IE proposal which mainly deals with the four different schemas (relational, hierarchical, network and object-oriented).

Wrapping facilities are only required if inserting new record to legacy systems is not possible through IE or to assign constraints of the information source in the IE. Wrapping is not flexible in terms of using part of the information source and could not fully managed if different users to access parts of the original complete information source.

Chapter 6

Application Design Issues in the Interoperation Systems

The majority of the existing database applications running over the Internet are mainly designed to work on a single network atmosphere. Also, the distributed applications, where replications exist for some parts of the data, are also replicating the code that enforces the constraints to maintain the data integrity. There is still no solid standard supporting database applications having heterogeneous schemas willing to interoperate and/or cooperate with other information sources across the Internet. On the other hand, there are number of requirements to be considered in any database application that wants to interoperate with other distributed heterogeneous information sources written using different schema types. Such consideration needs to be undertaken during the database application design phase. This chapter discusses the prerequisites to be exist in the information sources applications willing to interoperate with other unknown and unpredictable information sources in the light of the different requirements in terms of constraint enforcement and security requirements considered by the different database schemas. So, this discussion could be considered as the start for a larger complicated research to be undertaken soon by the research realms.

In general schema constraints are considered as axiomatic that cannot be changed or overwritten. Changing these constraints means simply alteration to the design of the application that uses the schemas. This is mainly related to the relationships between the database system different schemas, which are mainly derived by individuals different database management systems.

Unlike distributed systems centralized systems have all the related components locally. Components include the data, the database management system, the memory, and the programming language are specially designed to access the data. Distributed database applications are normally surrounded by number of layered environment. The DBMS where applications are running under them considered the first layer. The local operating system and the communication protocol also surround the DBMSs. The DBMS normally depends on the local operating system to take over the distribution behavior of the database application. There is absolutely no deference in designing the local and the distributed applications from the design point of view. While if the design of the application considers the distribution nature of the data then the database application will certainly act better. For example, the different constraints in the application suppose to be as loosely defined as possible so that they can be called as individual entities, which ultimately will require a slightly different design to be considered for the database application source.

Another preparation step for the interoperation process is the breakdown of the relations has to be studied carefully for the purpose to end up with the most suitable relations ready to interoperate with the outside world. A study to the expected read/write hits to the local databases by the global users should derive the set up of the system relations. The interoperable application should also be designed taking into consideration the categories/types of the distributed users accessing the local databases. This chapter presents a study to clarify the design requirements for a database willing to interoperate in a secure atmosphere with other schema type databases through the use of the *Interoperation Engine* IE facilities [Ashir2000]. The IE is designed taking into consideration the treelike relationship between the different database components such as the database, the tables, the attributes and the users, for all the known structured schema types.

Also, as a party willing to interoperate my information sources with others I am assumed to make the design of my information sources according to a standard that is understandable by the others. For example, things like the primary keys breakdown; the different naming conventions; the different required lookup table shapes and meanings has to be according to a standard that is followed by everybody. Of course, the existing standards on this regard do not include such low-level commitments and such need is considered as a massive task. The start is to include standards about the different attribute sizes, types and the other constraints related to the attribute. In addition, the likely recommended contents for the various information sources (i.e. tables in relational or class in object orientation) so that it is made known to others whom may be interested in the subject of that information source.

The IE is designed mainly for incorporating database-wide scalability and evolution support, as well as, component composability into an interoperation framework. In an environment where multiple information sources are dynamically connected to the Internet, it is highly desirable for users to be able to work with other remote data sources and treat them in their local application as if the data sources are local and from the same type. The IE assists scalability, evolution, and autonomy of the joining information sources.

Different users or different sites may access local databases and group of users may belong to number of sites and a single site may have number of user groups. For this reason my local database should be designed in a way to deal with all those access types. Additionally, the various type constraints in general should be designed in a way to accommodate the services for the entire single users and user groups. The design of the constraint should be done in way to fit with the local IE requirements. Also, as basic IE requirement the design of the screens should totally be segregated from the data entry operation in a way they can be called whenever they are required as, for example, objects or subroutines or in some cases values substituted within the IE knowledge base.

Lookup tables are considered as small tables describing small number of cases created mainly to facilitate the data entry and to avoid redundancy within the information. In most cases, the lookup tables that are created by data owners are by default unchangeable. In that the user who creates the lookup table doesn't al the time willing to open this kind of tables for users alterations. Therefore, in the light of information cooperation this point is considered by the design of the IE prototype.

Checking values related to the date and time in the IE is considered as an issue. Prior to checking this type of data it is assumed to consider whether the checking throughout the running database application assumed to be done according to the local source system time or the remote target system time. The initial assumption for the IE is that the checking on date values will be done according to the callers' system date. Also, checking on attribute values can be done either across the source database values or across the destination database values or even across both. This mainly depends on the information policies assigned by individuals and applied by the distributed IEs.

Validation operations on certain attribute value could be fully controlled locally by each IE. Checking for example the uniqueness of certain attribute value could be applied in all the cooperating databases or it could be done only against certain databases as stipulated in the IE knowledge base. This control assumption is applied in each local IE for the local database, which is responsible for the correctness of the local operations.

Additional benefits on the software development have been gained from the client/server development philosophy, which became the most appreciable technique that has ever simplified application developer life. Such technique has improved rapidly the process of writing a shareable application with having the information sources only lays over known places. The term client/server is used in conjunction with the database management systems if the application runs physically on one machine called the client, and the data storage and access is handled by another machine called the server. Such technology improvements have made the era of the specialized servers. Also, client/server architecture is considered as the technique of activating both the client and the server in a cooperation manner to complete a task.

6.1 Query processing in the IE

Queries on multiple heterogeneous databases are handled by the related IEs when cooperating with each other. Query uses the naming convention and terminology used by the information requester's database programming language. This process makes the mission easier for the query writer by not aware of the differences in the naming conventions and the other underlying information sources schema types. When the local system's database application program submits a query the IE starts its main process. From the query nature the local IE decide about the size of the data migration that should be applied by the two cooperating IEs to the local database profile. A full migration to the cooperating databases is only applied in cases such as reporting which normally requires the full database to be present.

Through the local IE the database a user will be able to build up the metadata of the database profile he/she will access by the local application. At the same time the database profile may be consisting of databases from multiple sites including the local system database. The databases may be heterogeneous in nature. The assumption here is a source relational database cooperating with remote hierarchical and object-oriented databases as shown in Figure 6.1. The final database is assumed to be relational database unified with the other two heterogeneous data sources.

The normal process for opening a database is first to open the database that contains the tables and than open the individual tables in that database. So, first the COMPANY database is opened, the EMPLOYEE table, DEPARTMENT table, and continues until the DEPENDENT table is opened. The local application normally prepared according to the local databases. When cooperation between the local database and other global databases is required only in this case the IE is used as a mediator. By using the IE many operations could be easily applied by which the security is one of them. Other facilities such as the interoperability between the local database and the other heterogeneous databases could be gained also. When interoperable cooperation is to be done between the source database and other distributed database the IE is activated by the application with passing the name of the database profile.

After the IE receives the activation of the cooperation process for the COMPANY database it prepares the database profile, which contains only the metadata structure. The local IE knows the different cooperating data sources. The attribute equivalencies are defined to the database profile that will be used by the local database application. Furthermore, the IE will prepare an image of the access privileges on the COMPANY subsystem. So, if many distributed users access COMPANY database, then the first user who will call will create the access view, which is a policy similar to current proxy servers catching techniques. By the viewing technique the security level in the IE could be increased by making sure users do not exceed their access limits. This operation in the IE is considered as a joint effort operation between the metadata subsystem and the security subsystem.

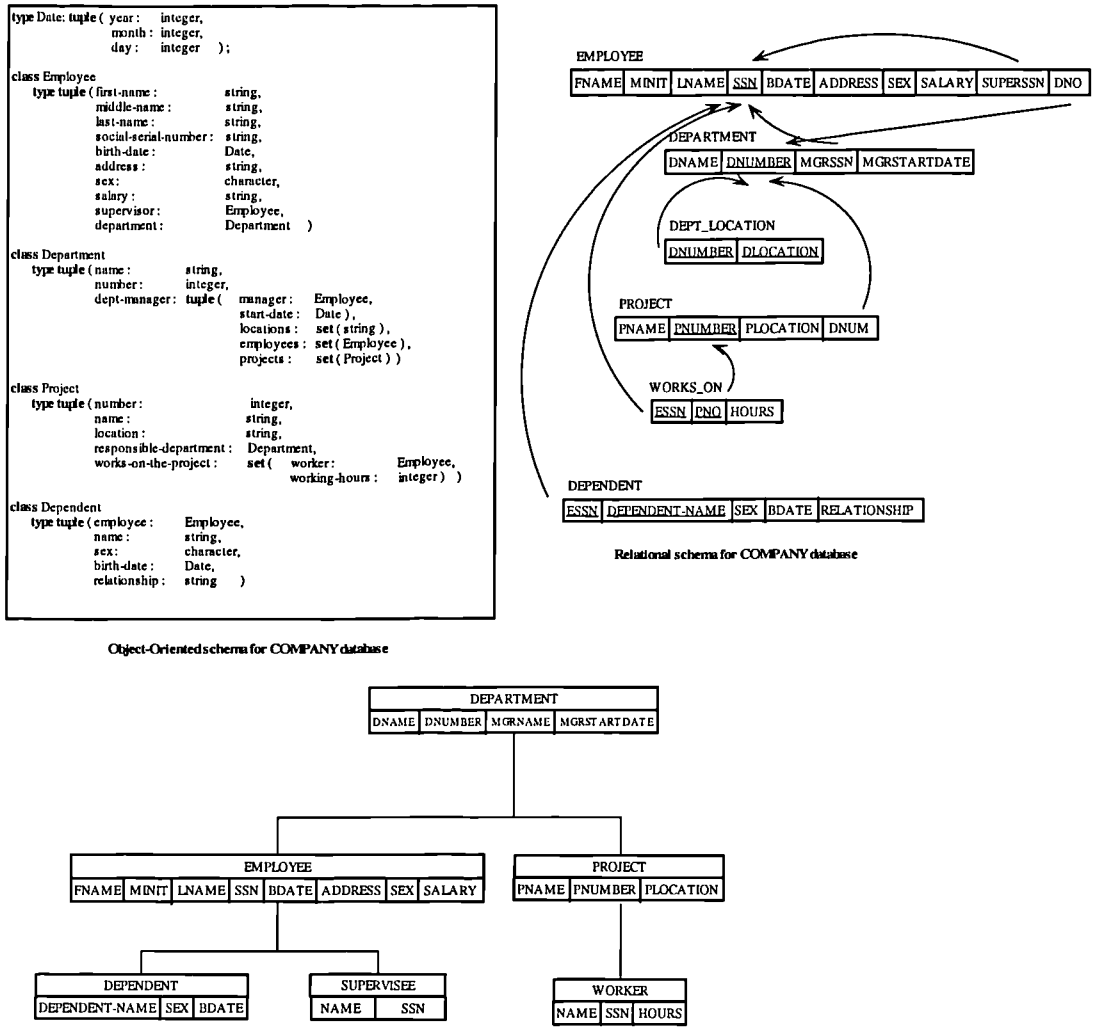


Figure 6.1

In the past example, the local database is relational and the two targets are hierarchical and object-oriented based. After the local IE creates the access profile, which is a copy of the relational database metadata structure, it will start making the decision about what records should be migrated. Here, in the example the proposal is a full migration to the three databases would be applied. At the beginning all the local relational tables will be migrated to the created profile. Then the IE, where the hierarchical database works under it, will apply three steps operation for the migration of the hierarchical records to the relational database domain, which is the requester database. The first operation is the definition phase of the metadata of the hierarchical database that will receive the hierarchical records one at a time, which is considered as the interface socket that will pull data from the requested information source and will migrate data backwards. As shown in Figure 6.1 above, this operation is considered as the definition for the mediator that will take records from the hierarchical COMPANY and will transfer them as one record at a time to the caller database profile. The hierarchical record looks like a single record as shown in Figure 6.2. The structure of that single record according to the hierarchical tree read operation regulations.

Figure 6.2

After the hierarchical COMPANY IE finish this preparation, then it reads the hierarchical database one record at a time and send it to the requester, which is the relational COMPANY, database. In this case, the IE in the hierarchical COMPANY side will require the existence of the routine which will reads the database one record at a time. Accordingly the hierarchical COMPANY IE will send the records one-by-one to the relational COMPANY IE, which will take the record, break it down according to its relational tables, find out the operation between the two tables, and update the relational database profile at the requester side.

At the time of checking the operation between the two tables IE will distinguish between the operations to guarantee the result accuracy. If the remote database records are to be unified with the source database records, which are the major used operation, then this will work by receiving one record at a time from the remote database. But, in case the operation is an intersection then this will involve reading the entire remote database and make it ready in the source database premises so that the intersection operation can be applied faster. The intersection operation between two databases will involve the existence of both tables prior to applying the operation because the less number of records table should scan the table with higher number of records to save time. The algorithms defined in Figure 6.3 make the different processes between the different distributed cooperating databases. Such processes are the unification operation between two non-balanced tables at one time, the intersection and difference operations.

For cases such as the intersection and difference operations handled by the IE as shown in Figure 6.3, the re-looping process is applied on the remote table records. The only difference is when the record is found in the intersection process the record will be kept in the source table and otherwise it will be removed, which is exactly the opposite in the difference operation.

(a) Intersection Algorithm.

```
Append contents of the original Source Table into the database profile
Do while not EOF Source Table
        Put Current record into Source TEMP

        Open Remote Table
        Do while Current record not = Source TEMP
                If FOUND
                        Keep the Current record into the Source Table
                        Exist this loop
                Else
                        Remove Current record from Source Table
                End If
        End Do
End Do
```

(b) Unification Algorithm.

```
Append contents of the original Source Table into the database profile
Do while not EOF Remote Table
        Put Current record into Remote TEMP

        Open Source Table
        Do while Current record not = Remote TEMP
                If FOUND
                        Exist
                Else
                        Add Remote TEMP to Source Table
                End If
        End Do
End Do
```

(c) Difference Algorithm.

```
Append contents of the original Source Table into the database profile
Do while not EOF Source Table
        Put Current record into Source TEMP

        Open Remote Table
        Do while not eof Remote Table or not Found
                If Current record = Source TEMP
                        Found = True
                        Remove Current record from Source Table
                Else
                        Skip one record in the Remote Table
                        Found = False
                End If
        End Do
End Do
```

Figure 6.3

Usually when preparing the access database profile, which is basically the tables view, the profile will initially contain the local records that will be migrated first to that profile in cases like unification. Also, from the schema cooperation profile table, records from the distributed cooperating data sources will be added to the profile. By first applying the operation between the initial database profile, which initially contains the local table records, the cooperation process will continues following the sequence link in the schema cooperation profile until the completion of the last cooperation between the database profile and the rest of the cooperating tables [Ashir2000].

6.2 The different schemas components security subsystem in the IE

Security implementation could take two distinct ends. It could be implemented as the highest and the cost of this security strength is the performance or as the minimal and the performance is too high with the cost of nothing is secured. For this purpose it is recommended by security specialists to keep security in the middle so that the system is considered secure and the performance is acceptable [Castano95]. This section is mainly discussing the access right assignment part in the IE which is the security part dealing with only the type of access given to the different users. This is a step further by making use of the tree like relationship used in the implementation of the IE by no means of comparing this approach with the different database security standards.

The IE considers all the databases as objects and have to be protected from the local and global users. As a design matter perspective, the security subsystem in the IE is totally segregated from the other subsystems. This is due to the security level the IE is designed to achieve. Because different users maybe assigned different access rights on different systems, the segregation of the security information will assist in accomplishing such facility.

The security issue in the distributed databases owned by different people is complicated and not straightforward as in the centralized database systems. The cooperating database systems may not have similar access rights on certain data or may not be treating the users similarly. Assume that the local database, which is the relational COMPANY, permit full access rights to the local users and varying partial access rights to the global users. On the other hand, the other two cooperating databases, which are the hierarchical and object-oriented COMPANY, do have some restrictions on their databases when accessed by the global users. The policy of the IE security is to follow the rules of the database owner.

There are number of security assumptions to be undertaken and resolved by the local IE. One assumption is the addition of new records on certain databases could be only accepted by some of the global cooperating databases. In this case the checking on the uniqueness of the entered record from the local database application is suppose to be done against both the local database records and the global site records. As an example, assume the hierarchical COMPANY schema shown in Figure 6.1 accept addition of new records by certain user group. In this case the local IE should have a definition by which to add new records on the remote hierarchical company and whether or not to add the same record to the local relational COMPANY or not. This kind of decision is mainly handled by the IE security manager according to the object-oriented classes structure as displayed in Figure 6.4.

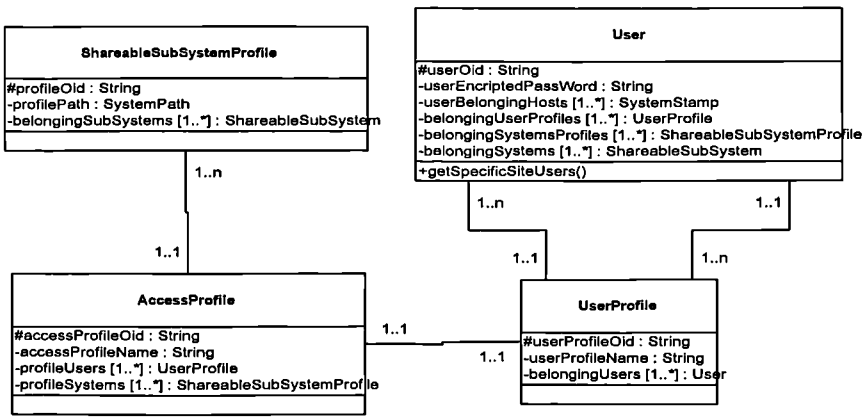


Figure 6.4

Figure 6.4 illustrates the concept of inheritance between the three classes is valid in the IE security management. The lower class, which is always carrying the one-or-many side in the above figure, inherits the access rights of the higher class. For example, if a read operation is only permitted on certain subsystem, then by default the access rights on the belonging schemas and attributes of this subsystem could have read access or lower.

Constraints forming an important part of the database model security. The referential integrity is measured based on the provided constraints by the model. Schema constraints are considered as a constitutive matter defined by the database management system model that cannot be changed or overwritten. Changing these constraints means simply alteration to the design of the application that uses the schemas. This is belonging to the relationships between the database system different schemas, which are mainly derived by the individual database management systems. The Interoperation Engine prototype security manager is considered as an additional security mediation layer considering the reassignment of the original security given by the original information source database management system. Hence, the IE security assignments has to be derived from the original information source security assignments. For example, if the original information source database management system assesses only read access on certain schema, than this constraint cannot be reassigned by the IE security manager to access such as write on some of the attributes. Section 6.3 will explain further the breakdowns of the security assignments on the shareable subsystems in the IE.

6.3 Sub systems constraints

The IE security manager is designed to support two assignment types. It could be either defined as many systems uses a single constraint or a single system uses composite constraints as clarified in the UML class design as shown in Figure 6.4. The following are the possible access assignments on the schema level.

- All read with some exceptional hides and writes on the schemas and attributes
- All read
- All write with some exceptional reads and hides on the schemas and attributes
- All write

Additionally, some information sources owners don't want to let other than specific number of users to access their information source simultaneously. This is due to the quality of speed (i.e. the load control) on storing and retrieving of the information parts required by the information source owner or other security restrictions such as preventing group of users profiles from accessing their information source.

Also, as for the future planned work, which is considered as an added of the constraint, is to add the query or service type to be processed for the users according to some criteria related to the level of the user accessing the information source. This criterion should indicate the level of the user and accordingly his required service will be fit in the necessary queue. Each criteria type will have a separate queue that takes a time slice on the service processor. The different user categories are defined through the *GolabUsersCategories* and *LocalUsersCategories* classes respectively.

The IE will take the responsibility of creating the different scripts of the different access flavors for the distributed users on the distributed information sources in a dynamic manner using an XML like language to define the various access profiles prior to the data transportation process between the different distributed IEs.

6.4 The access view by the IE

As a single site provision, this site might have a number of shareable subsystems where different groups with different access rights accessing them. One of the main objectives behind the IE puts the access assignments on the different shareable subsystems under control. By querying the IE it will be possible to exactly identify the access rights on certain subsystems or for certain users or group of users. As identified by [Ashir2000] users are defined to different IEs as two levels tree where the first is the site they are belonging to and the second is the users themselves. The site that will give access to users may group them again according to some internal site policies such as allowing only certain number of users

from each site or according to the users occupation. When users are given the access on the information source by the site that owns the data, this site adds a new level to the tree, which is the group. The levels of the tree are the group defined by the site, the different cooperating system stamps, and the users defined under each system stamp.

After defining the subsystems and the access rights on them, the IE is ready to make the access view for the users whom will access the cooperating database systems. At the beginning the access will be created into two main parts. The first is the list of the user group. The second is the access rights to the subsystem. The first list, which is the users group, is the defined user groups in the local site. Those are consisting of the users given by the global cooperating systems, as well as, users of the local site. The access rights are the predefined access privileges given by the data owner plus the reassignments of the access privileges done by the site where the access operations on the cooperating data sources will take place.

The access view of the users on the cooperating databases is completely controlled by each local IE. It could be defined as a site wise, subsystem wise, subsystems profile wise, group of users wise, or as users groups profile wise. The access view could either be created for a group of users or a reassignment of the access privileges could be also generated for the individual users in that group. In the first case two tables will be generated where one contains all the related users and the second which is a single record table contains the database system related tables and their attributes accompanied with the access right on each of the database components. In the second case if the local IE DBA generate the whole individual users access view this operation will replace the group access view which is originally consisting of two related tables with a single table consisting of all the users and their access rights over certain system.

In case the view definition is set up as a site wise the user from that certain site who will make the request first will make the complete view prepared by the called IE. All the related users to the cooperation are going to be included in the access view. Normally IEs receives users as a list titled by the site information they are originally belongs to it. This access view will require information about each individual user from that site specifically, information about the systems the users are involved to, and information about each of the systems' involved tables and attributes. In some situation the local IE DBA may define certain user group consisting of users belong to different sites. In this case additional information about the original site of the user is added to the previous information list. This situation is also applicable to a group of users wise and users groups profile wise. The opposite operation is applied on the subsystem wise and the subsystems profile wise. This case involve getting first information about the cooperating subsystem and then include the information of all the local and global users accessing this subsystem.

The final access view will be an amalgamated view of the pool each user should have access to it plus the type of the access right to each of the database elements up to the field level. The access view consists of data such as user related information, accessed subsystem information where part of the data is the same as the one defined in Figure 6.2, and an access right on each of the database elements. The process of converting the related records to a single non-normalized record runs on all the schema models. This process will take over the burden of checking the validity of the data model constraints. Also, it will assist the migration process between the heterogeneous data models.

After the access information views are being created each of the IEs will take care of the optimization process on each of the access views. For example if the number of users having similar access to the same database then the optimizer will split the access view information into two tables. The first will carry the information concerning the users and the second will carry the information of the access to the database. By reaching this step it means the cooperating data sources are converted in a simple form that is understandable to all the users. After this stage comes the migration of the flat non-normalized data model to the distention data model where all the process is completed for this final model. To clarify this point as the example defined in Figure 6.1, it was assumed that the host schema type was the relational COMPANY and the first target database schema was the hierarchical COMPANY. Now, let us assume that a unification process is between both schemas. After making sure of

the validity of the user and the query, the transfer operation of the records will start by sending one record at a time to the source database storage. Since the operation is unification between the databases then according to the algorithm defined in Figure 6.3(b) the new records will be migrated to the relational COMPANY. Each transferred record from the hierarchical COMPANY will contain the contents of the six relational tables at the relational COMPANY premises. As known from earlier explanation, the cooperation operations are applied at the database level. At the table levels the information consumer will be able to update only certain tables and blocking the others from update according to the business requirements. In the case of the example shown in Figure 6.1, relational COMPANY the data owned might only need to update the EMPLOYEE and the PROJECT tables and leave the others as they are.

6.5 Schema design constraints validation in the IE

The schema design constraints are normally enforced by the DBMS either when reading from the database or when writing to it. In the case of reading from any type data model the IE will attend a de normalization process to simplify the mapping operation between the non-equivalent and different data models. On the second case when writing to database the IE has no option other than depending on the wrapping encapsulation technique [Mowbray95]. Especially if the database application was built on legacy database system then adding new record has to be done through passing the record values as parameters if the PL of the database system allows such thing. Also, the IE is designed to assist in simplifying the writing mission. Different constraints could be gained from the ones recorded in the IE knowledge base before doing the actual record write in the actual information source. In this case the wrapped code part should be simple and straightforward.

To properly encapsulate a legacy database application PL with the assistant of the IE, (1) it must appear to the application requesting such service that it runs in its local environment, which is location transparency. (2) If, for example, an application expects an initialization file of a certain name to be in a certain place, the IE would make sure it is there prior to applying the actual wrapping process. (3) If the application will write a temporary file in a specific place, the IE would ensure that there are no conflicts in terms of process replication. (4) If other application processes needs to be initialized and started because the target legacy application requires them, the IE would do this preparation. This open-ended sort of management is the source of considerable concern to those tasked with implementing wrappers. The IE would do most of the checking using its pre-assigned knowledge in a step forward to simplify and minimize wrapping required.

Consider an application that performs a simple data transformation. It reads an input file in a certain format and writes an output file in possibly a different format. Simply to run such a programme, a wrapper needs to take the following steps. (1) Ensure that there is no file naming conflicts. (2) Create and write the input file in a specific format. (3) Invoke the application. (4) Monitor and check completion status. (5) Read and parse the output file. (6) Remove the input and output files. (6) Return the result. Even this trivial example requires a number of actions to be achieved. It is easy to project the amount of complexity involved in robustly wrapping a moderately complex application. The IE will contribute in the checking stage of the different constraint prior to the actual record reading the will involve the wrapping of the code that will only write to the information source. All reads constraint management will be done according to the IE knowledge in each of the sites and will not require the actual application intervention.

Constraint validation in the schema design of the heterogeneous databases is considered as one of the most difficult areas of research in the database field. The design constraints are simply reflecting the relationship between the different database records. Therefore, as an indispensable requirement, enforcing the different design constraints involves the knowledge of the schema design building. For example, approaching other than the root level record directly in the hierarchical schema is illogic and should be rejected.

It has been assumed by the IE initial design that the information producers' IE will be responsible for managing any of the constraint assignments defined by its domain database,

which will be considered as application design inputs. This step would be achieved after preparing the necessary design of the database application that will interoperate with other distributed and at the same time heterogeneous databases. The IE will receive the constraints that will derive the different requests on that information source.

As for the example given in Figure 6.1, assume that a remote site is only interested in knowing the current running projects where that remote site uses a hierarchical schema COMPANY database. The local IE of that schema will traverse the hierarchical COMPANY database and will extract only the records related to the query. Of course, the databases own programming language through its local IE will only handle this process. The local IE will first make sure of the compliance of the users' query with the schema design constraints. In this case, the sequence in reading the records would complement the hierarchical schema rules plus filtration rules if they exist.

This chapter highlights the requirements of the different schema interoperation in a manner to make the mission of information cooperation easy so that the information consumers can easily understand and merge them with their own information sources. It appears that there are some other requirements to be met for the distributed heterogeneous schema cooperation such as information reading and writing sequence in the information source. Although, the heaviest involvement is to make my local database source application segmented in a way to make the necessary cooperation with the other data sources and at the same time retaining all my information source constraints. This requirement is for cases that are not possible to register the constraints within the IE knowledge base.

It appears during the study that reading operation from the distributed heterogeneous schemas while retaining the data source constraints can be applied much easier than in case data source accept new records or updates. In that case the data source will need extra checking over the written records and this means a technique such as the wrapping or equivalent is required only in case constraint cannot be checked through IE knowledge base. The main purpose of the undertaken study is to simplify the heterogeneous information sources interoperation process with retaining the autonomy to the database owner. Even a consideration of the differences in the database design of a similar schema type by the IE has. Therefore, the prototype design mainly depends on the tree like relationship of the different database components. The database interoperation research is an ongoing task. As an extension to the work an initiation for a database interoperation standard that will stepwise the design process of the new database applications willing to interoperate with other similar and different databases will be discussed. As well as, a consideration of including the legacy information sources in the interoperation process with no changes or minor additional steps will be undertaken.

Chapter 7

The Interoperation Engine System Architecture

The rapid growth in the number of users of global networks such as the Internet, the number of systems used by those users, the amount of the available information, and the growth in the bandwidth has made new demands towards finding new techniques for interoperating between the enormous information pools that already exist. Many people from companies, government, schools and individuals will have access to an increasing amount of data. Indeed, users have already started to access huge amounts of information from disparate sites. Thus, the problem typically associated with (1) the size of the available data, (2) the heterogeneous nature of the data, and (3) the extraction of the necessary information that will form the knowledge which will facilitate the interoperability of the distributed heterogeneous databases.

The first problem, which is the available size of information sources, is caused by many factors. The main one is the rapid connectivity of the local area networks, which forms the global backbone of the public Internet. Moreover, the success of linking the three-tier architecture (mainframe, mini, and Local Area Networks) has enlarged the size of the problem. Instead, this has increased the number of databases becoming available over the Internet. Also, the advances in linking heterogeneous communication protocols maximize the size of the available information source and thereafter the problem.

The second problem, which is the heterogeneous nature of the information sources willing to cooperate, is mainly caused by the ongoing demand in the field of information presentation. The new powerful platforms that have emerged with the new data models such as the object-oriented which support voice and video has changed the view of the new information requirements. The situation ends up with new powerful information platforms coexisting with old legacy systems using hierarchical, network, and relational data repositories. Legacy systems cannot be discarded for many reasons, most of which relate to the huge amount of information stored on them and the cost of translating and moving them to new platforms. Sometimes, even if the data could be migrated to new platforms, the rewriting of the code is an almost impossible process, and would be too costly. Additionally, the representation of similar information may differ from one place to another, which may be caused by the differences in thinking between people. Also, the representation of the information is derived by many factors such as the data owners' business requirements and other related to the data owner integration process of the new systems with the existing old systems.

The third problem is related to the information required by any one to be able to join the cooperating data sources, so that the interoperation is done in a transparent manner, which is a major issue. No one person can be considered familiar with all the different data sources and that he may access only the information relevant to him. Such process considers that person has the required knowledge about the information space of his interest that he can get from the Internet. Also, he should have a way by which he is informed about any changes in the information source he is already using. This is to stays up to date and gets the required services from the remote information repositories when a change occurs. Because of such difficulties, the problem related to the mechanism of dealing with the rapidly increasing sources of data is significant. In order to cope with the rapid increase data sources environment there should be a solid mechanism for gathering only the necessary information about the cooperating parts of the distributed heterogeneous data sources. Among the huge amount of data available over the global network a very small amount is of interest to

interoperate with other data sources. As long as this problem is not being dealt with, its breadth will become uncontrollable. Having the required knowledge concerning the cooperating databases is the major solution to the problem as a first step.

The approach to cooperating databases should ensure no bad effect on other normal services offered by the database management systems. Among those services are: schema translation management, programming language translation management, semantic inconsistency management, and other aspects related to the operating system and the communication protocol layers which are considered as outside the database research realm.

All these issues make a strong demand towards having a new dynamic mechanism for the cooperation of the heterogeneous distributed data sources. This issue needs to be looked at in a new way. Although there are currently a number of suggested static definitions for the cooperating data sources, no one is considered as the ideal solution for the rapidly increasing number of cooperating heterogeneous distributed data sources. Disco project [Tomasic96a] is one example of the static definition of the necessary databases interoperation parameters and which our view is considered as a dynamic implementation of parts of the Disco.

A middle engine that can act as an added-on facility on the Internet browsers, responsible for binding only these heterogeneous distributed data sources of interest, has been proposed. This middle engine will not require the set of related applications to agree on one global view. The approach is based on the information availability and information demand. It will also depend on the information advertisement technique for the purpose of advertising an available information source. The infrastructure will also form the foundation to buildup all the supporting areas in the distributed databases (i.e. security, transaction management, history tracking, etc.)

7.1 The Proposed Approach in Brief

The current attention in the DBMS area is towards the development of tools making possible the coexistence of different DBMS supporting various data models. This direction requires methods of equivalent data model transformation and methods of constructing unifying data models and languages promoting generalization of various approaches to the development of DBMS language [Kalinichenko90]. Because of this, a number of projects have been established in this area such as the Jupiter System that is considered as a prototype for multi-database interoperability. In this system, the majority of the work has been done on the mapping mechanism between the different schemas of the databases. The mapping of schema has been considered between autonomous and possibly some heterogeneous databases [Murphy94]. Also there are other recognized projects in this area such as IRO-DB and ESPIRIT III. They are providing a method of integrating heterogeneous data sources from the design and the data perspectives. They allow for the integration of heterogeneous object-oriented and relational DBMSs [Busse94].

On the other hand, the proposed approach reserves the autonomy fully to the data owner. The data owner is the only one who has the right to advertise about the shareable parts of the data he owns. He is also the only one able to give certain users access to certain parts of the data. The approach mainly deals with the heterogeneous data sources in a balanced and unified manner.

The knowledge cooperation database in each of the sites is supposed to work as a custodian process for all the cooperating databases, which are originally under the control of multiple organizations. So, central control of the cooperating databases, which usually centralizes the autonomy process through the creation of global schema, is not required by this approach. Instead, the cooperating data sources will be loosely coupled by the distributed interoperating knowledge of the proposed prototype. For the purpose of information sharing the IE support two different approaches for information cooperation. They are the decision about the shared parts of his/her information and the advertising about this information space.

In the first approach the data owner will decide about the parts of the data he owns and he wishes to share with others. In this case the data owner should inform the others with the

necessary information about the data space he wishes to share with them and the way they can get it so that they can share it with their own information sources. The data owner will be able to specifically record the sites or even specifically the persons who have the right to share his data. In this case users at any site will write their applications across the data in the local cooperating knowledge, which will be responsible for the further related bindings with the local and global data sources by using the IE knowledge base. Here in this method, the data owners will decide about the commonalty in their data with the other remote data sources.

In the second approach the data owner will send an advertisement to all the cooperating data sources asking either about the existence of certain information or clarifying what locally exists and asking for information cooperation. The idea here is to make an atmosphere that is similar to the current browsers. In that a specialized database search engines responsible to get and connect with global information sources is designed. The advertisement for a shareable information space will be propagated to every IE registered in the server responsible of keeping the list of the cooperating information sources. Here every IE means only those I am willing to inform about the information space I got for share. The registered information sources addresses should exist at a number of sites where each backs up each other. By this technique we are building up incremental information source knowledge capable of handling the wide range of different existing knowledge areas.

The following example is to clarify what has been mentioned above, Suppose a company ABC is specialized in importing products and interested in new consumable products. Assume it is currently getting information from seven companies manufacturing products where company ABC imports from them. The seven companies are known from the knowledge acquired by the ABC interoperating system. ABC also may send a request for all the other manufacturing companies asking for certain information. In this case if ABC gets a number of other replies then they will be able to link the new suppliers with their list and thereafter to the existing information sources so that all will act as if they are a single information source. Here ABC queries are written across a profile in their interoperating knowledge rather than directly to the distributed and possibly heterogeneous data sources. In this case knowledge expansion is considered as an incremental dynamic process rather than a static process, which may need many interventions until changes, gets work. The idea here is to have a specialized database browsers and mergers to setup the required cooperation between any information producer and any information consumer.

The main advantage of such approach is to create the cooperation atmosphere between the different information sources in an incremental manner. Moreover, this cooperation will not involve changing the existing compiled and tested programme source. The only changes that will be taken automatically by the site IE are those done in the local interoperation knowledge, which is the confirmation of the addition of new cooperation information sources. Most of the cooperation processes in the IE are optional so that the user can set and reset whenever it is required.

The overall approach is based on the advertisement of certain information sources and the information cooperation requirements of individuals. However, for such an approach to be feasible, cooperating data sources must give the proper authorities for others to cooperate in a manner valuable for both. In particular, such an approach supports both incremental and quick cooperation between the heterogeneous distributed information sources.

With this in mind the Interoperation Engine IE has been designed, which is a combination of information about the cooperating data sources and routines bounded with route maps and work flows for the different operations during the lifetime of the request. The information about the data sources (metadata) is the part responsible for the heterogeneous data sources linkage, as well as, the constraint enforcement of the different schemas. Additionally, the IE contains a subsystem responsible for keeping the necessary information about the different distributed users in the cooperation domain. This subsystem is considered as the first firewall against unauthorized access to any of the cooperating data sources.

The major concern is to build a system, which is easy to use, as well as enabling the user to use the programming language to which he/she already accustomed. Also, the flexibility in the

number of data sources that can be added to the cooperation is done in the background without any additional requirements from both the data sources users and the application programmers. The aim is to provide a fully friendly system with an easy interface so that the DBA, who will be responsible for the bindings with the remote sources, has not much to do to hit targets.

7.2 The Proposed Prototype Architecture

For the purpose of supporting the control of information interoperation and information cooperation, the IE prototype has been designed to provide three main groups of services:

1. Proper information gathering about all the cooperating interoperating information sources and all the related users including the management of the information sources.
2. Management of the incoming requests to the local hosted information sources through the local IE.
3. Management of the outgoing requests from the local to the global information sources through the local IE.

Figure 7.1 shows the overall architecture of the IE system. The main work is conducted in the cooperating local/global user management and the cooperating data sources knowledge management. The IE design also facilitates the achievement of the work carried by both the interoperation facilities manager and the general purpose DBMSs facilities manager.

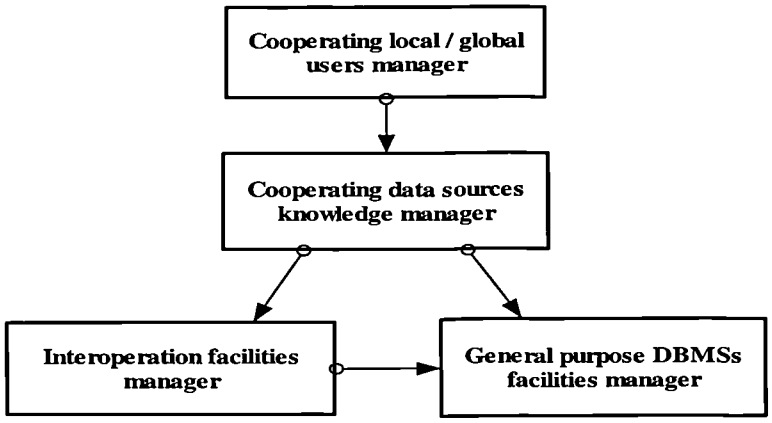


Figure 7.1

The gathered information about both the cooperating data sources and the users will be mainly categorized as local and global data sources, as well as internal and external users. The main role this part plays is the linkage between users and approved data source links. Further, It will facilitate the workflow of the other two services.

The IE proposed solution has many interrelated components responsible for handling the different areas of interoperation. The connectivity of the different components is shown at figure 2.1 where each of the components provides services within its area. It also shows the preliminary design of the overall IE system including the flow of the different services in the system. These services form the four categories shown in Figure 7.1.

The preliminary proposal is to make the IE operate in three interrelated stages:

Stage 1: defining the different shareable components on the local IE repository.
State 2: advertise about the shareable information space, which consists of two sub-stages:
Stage 2.1: the selection of the shareable parts of the total information space.
Stage 2.2: the selection of the user groups whom can use the local provided information space.

Stage 3: define the links between the local and global information space that will be used by the local applications using the local IE knowledge in this regard.

Figure 7.2 provides an explanation of the first two stages.

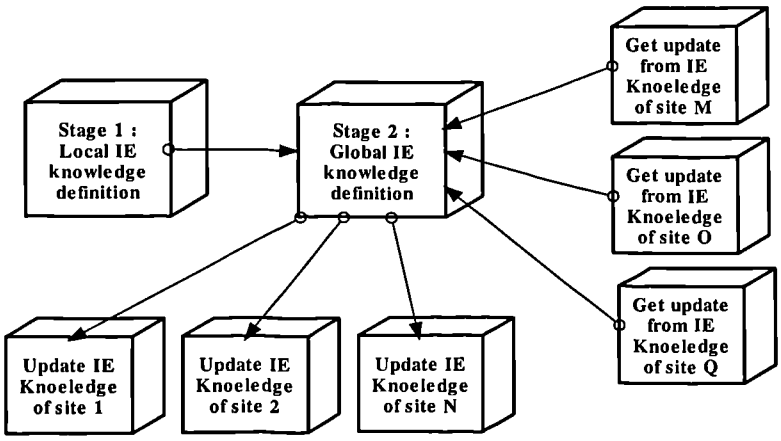


Figure 7.2

The local IE knowledge definition contains the definition of the whole local systems. Only the shareable parts to the global users will be traversed to the global IE knowledge base side. Here the local shareable IE will contain the necessary local and global information to bind the cooperation data sources. At the same time the local shareable IE knowledge will get updates about the shareable information space from the global cooperating IEs. The third stage is meant to prepare the necessary linkage between local information sources and global information sources into a data sources profiling manner so that the internal applications can use those profiles rather than direct connection with the global data sources. The static definitions of the parameters for linking heterogeneous information sources defined by [Tomasic96b, Liu96] should benefit from the dynamic definition of this work. The advantage of this approach is that any change in the information space does not require changes in the application source. Also, the application reliability considered as high compared with techniques requires application source changes.

7.3 IE Operations

In abstract terms, IE consists of three main operations each of which plays a role in the success of user queries in manageable and secure manners. The three operations are the *incoming request* management, the *outgoing request* management, and the other *local services* such as the knowledge definition of the parts of the local cooperating schemas and all the other related necessary maintenance operations, which is the basement step for the interoperation scenario.

First, *Incoming requests* are the hits the local IE will receive from local and global requesters. The local IE will basically authenticate the requester, check the authority against the requesters' request and insert the request in the queue according to the requesters' level of importance stored in the security manager knowledge base. The other facilities such as the PL translation, query decomposition, transaction management and schema translation will only be applied on-demand when necessary.

The incoming requests will first arrive in the incoming request pool. From this point, requests are categorized into three main categories. The first are requests with high priority, which will have the highest priority in getting executed over the others. The second are the medium priority requests, and the last are the non-critical requests. This sort of categorization will insure that the IE system does not block an important request from either internal or external users.

According to Figure 2.1 the proposed prototype is supposed also to support requests management. Requests will be first collected in the requests' pool. The local operating system should handle the queuing process for the incoming requests. Tasks will then be handed one at a time to the first level IE checking process, which is the dynamic access manager. At this stage a decision should be made about which request is to be processed locally and which one should be forwarded to the replicated sites. The dynamic access manager, security manager, and the replication manager will handle this task. Once this operation is over, it will start checking whether it is necessary to apply any of the on-demand processes or not.

IE replication managers have two kind operations. The first is re-forwarding incoming requests to be fetched from another which is either local or remote duplicated data store. The second replicates local databases or part databases to remote data stores for security or load balancing purposes. Hence, local replication manager may sometime transfer requests to other replicated data store. This operation will be applied either for the purpose to balance the load on the local data store or some other security purposes.

Replication manager also supports two types of replication mechanisms. Those are On-line or off-line replications. On-line replication is the process of replicating records into different local and remote data stores in parallel so that any changes will occur to the main data store will be reflected to the replicated data store at once. This type of replication will insure all the data copies to be up to date. Off-line replication is the one applied after certain amount of time as a batch process or when activated manually.

Requests arrive from either remote or local user. After the user is authenticated, the request elements, such as the system the user requests from and the related schemas and attributes get authorized. As a further step the user ID plus the request data store path is forwarded to the replication manager to check whether there is a re-forward to the request for another data store or not. If no re-forwarding is recorded by the replication manager knowledge base then this means the request will be fetched in the same place. Otherwise, an update will be sent to the replication manager knowledge base, which has issued the request, so that any operation on the specified database will directly be re-forwarded from the requestor site. Once the update to the replication manager is done then other users have the same access rights on the data set will make use of this update and will get the re-forwarding directly from the site they belongs to.

Second, the *outgoing requests* will be first checked locally against the gained domain of the user who applies the request. This step will prevent local users from losing time and effort before their request reaches the remote site and get rejected there. The assumption here is that any changes in the access domain for any of the users in the cooperation will take place at the same time the change occurs and the user access space will be updated.

Third, in the *local services* first phase the user will start from defining the high level type of the schema he wants to create. In this case the user will chose between the four known schema types which are relational; hierarchical, network; and object-oriented. There are two possibilities, either the schema already exists, or it is new and will be created by this part of the process. If the schema is already exist then the IE will conduct checking to the entered data to match them with the original schema. Also, in this case the user will be able only to define the shared data parts by leaving the other non-shared data undefined. If the IE will create the schema then the user will be able to handle the creation process into number of phases so that the user will not need to setup the security access levels and the users profiles prior to writing the database application. Of course, he/she needs to create the set up together with other parameters such as the one related to data replication later after getting the application running and after setting the users' policies in accessing the premises data.

Record definitions in all the record based schema types are the same. Object-oriented schema definition also do not differ from others much. From a past study that has been conducted on the four schemas (relational, hierarchical, network and object-oriented) a conclusion has been reached considering that still there is no standard object-oriented

schema. Furthermore, the object-oriented schema seems to be the amalgamation of the remaining three schemas. From investigation conducted it is seen that the shape of the object-oriented database may looks like a quarter balanced tree in a way that the similar information could be seen as a distinct peace from within the tree. The only difference between the four schemas is the relationship between the different types. The user therefore can start by giving information about the schemas in the local system, which forms the local information space. At this stage the user will define the system, the related schemas, the related attributes together with all the necessary information for sharing data with other information sources.

Chapter 8

The Interoperation Engine System Design

There is currently a tremendous increase in the number of data sources that can be queried across the Internet. Many of these data sources are databases. It is most helpful if the data source is readable to the user who tries to access it. Additionally, the user needs to know how to get the information of interest from the database. Furthermore, in order to remain current, the user should update the data by downloading the database as and when necessary. To solve these problems, the design of a knowledge base that will define and automate the interoperation of the heterogeneous databases for web-accessible sources has been presented. The contributions are as follows: (1) the process of handling the necessary parameters that are required for establishing the interoperation operation between the heterogeneous cooperating information sources and at the same time used in establishing the mediator setup between the information sources; (2) designing the heterogeneous databases metadata capable proxy server, the search engine, and the other necessary client interfaces; (3) designing the access control system that links the different distributed users to the heterogeneous distributed information sources.

All database systems, regardless of schema type (relational, hierarchical, network or object-oriented), are mainly consisting of three building components. These are the system that contains one or more schemas where each may contain one or more attributes. This characteristic in data schemas may facilitate linking the distributed information consumers with the available distributed heterogeneous data sources. The manner by which this link is to be accomplished should be as transparent as possible so that the local user get the feel as if he work and access local DBMS already known to him. Current Internet browsers do not fully support the interoperability of the heterogeneous databases. As Internet browsers offer browsing facilities for users by gripping information or data from the entire Internet connected information sources this prototype is specialized only in facilitating and connecting local databases to the heterogeneous databases scattered over the Internet in a manageable transparent manner. Moreover, Web information management is one of the most important current trends in data management technology and research area [Dogac99].

In the prototype, which is the *Interoperation Engine* shortened as *IE*, the mechanism of propagating the component databases of interest to the global in a dynamic way so that distributed users know what available data sources they can access has been addressed. The purpose of the prototype is to simplify the interoperability of the distributed heterogeneous databases. Also, it will automate this process by giving the local users access to heterogeneous databases through the use of profiles by which they contains information of interest from more than one source. Also, in the prototype it is assumed that the database schemas are not reasonable stable and may keep changing, and in certain stage they may get excluded from a certain database cooperation accumulation.

8.1 Database Interoperability Requirements

Interoperability between heterogeneous information sources can be defined as the ability to generate a single virtual view of heterogeneous schemas without sacrificing autonomy. An important aspect of this is to coordinate changes of such autonomous schemas while providing guarantees on the quality of the different schemas information flow and the quality of the different services on the schema.

It is worthwhile to restate that the sentences data source, information source and schema will be used interchangeably to mean a database. Also, interoperability phrase may be used for

the hardware and for the software. For the hardware it means the capability of connecting different vendors hardware. In software it has many faces where the application and database interoperability are the most important. Reuse of the legacy applications and their attached information sources are the most frequently quoted requirements for application and data interoperability. The profound problems still challenging the area of software interoperation are the syntactic and semantic data interoperability.

Of course, interoperability between the heterogeneous and distributed information schemas still suffers from some of the above mentioned problems which have been limited by the new advances in the field. The new advances have shown that through the use of the proper mediators most of the interoperation problems could be overcomed. Also, the new advances in encapsulating parts of the legacy software have also an added value in the merging advances of the legacy and new database applications.

The heterogeneous database interoperability means openness of any type schema to any type schema for all the database operations. By the database operations it is meant add, delete and edit. Assume if an information source user uses a hierarchical database, he can do any database operation on any type remote database. In this case the word cooperation may be closer to the meaning we are working towards to achieve, which is wider than the normal interoperability. The most basic shape of cooperation is to have an information source, which cooperate with other heterogeneous information sources, provide similar information to form a single unified information source similar to the requester schema. In cooperation both the information producer and the information consumer should apply certain standard to achieve the interoperability, which is the discussion highlighted in this chapter.

The overall work in the area of database interoperability is nearly directed towards achieving number of strategies that are necessary for accomplishing the interoperability process among the distributed heterogeneous databases. The initial main strategies that have been considered during the present work are:

1. Initiate the strategy of metadata registration in the cooperating information sources that acts as the infrastructure, which will be used by both the information produces and the information consumer to advertise about the existence of certain shareable information.
2. Binding the local schemas with the global cooperating schemas and allow incremental addition to the cooperative module without any changes in the local queries using the IE knowledge.
3. The requirements towards providing database services in the current Internet browsers using specialized metadata management capable proxy servers. The relationship between the different components attached to the proxy as shown in Figure 8.1.
4. Provide the necessary management services for the interoperation such as security services; syntactic/semantic data dictionary services; history tracking and the replication services.
5. The initiation of a universal object identification strategy to provide the maximum flexibility to the interoperation objects that will be used from anywhere.
6. Design the infrastructure of the security management, which acts as a unified security manager serving the local heterogeneous information sources.

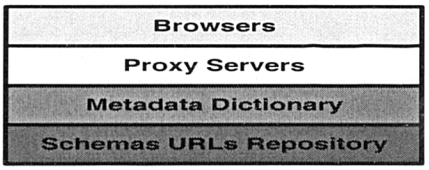


Figure 8.1

It is also obvious that the real major database interoperation problem is related to the information required by any one to be able to join the cooperating data sources, in a way to make the interoperation process done as transparent as possible. No one can be considered

familiar with all the different data sources and that he/her may access only the information relevant to him/her. Such process considers that a person has the required knowledge about the information space of his interest, which may exists overall the globe. Also, he/she should have a way by which he/she is informed about any changes so that he/she stays up to date and gets the required services from the remote information producers. Because of such difficulties, the problem related to the mechanism of dealing with the rapidly increasing sources of data is significant. In order to cope with the rapid increase data sources environment there should be a solid mechanism based on a unified policy by which both the information producers and consumers follow to update each other. Among the huge amount of data available over the global network a very small amount is of interest to interoperate with other data sources. As long as this problem has not been dealt with, its breadth will become uncontrollable. As a first step, having the required knowledge about the cooperating databases is the major solution to the problem.

As part of the proposed prototype a web-enabled database interoperation proxy server can be used to form a virtual community, where participants in remote locations can exchange metadata information about the cooperating information sources in electronic format. The existence of such facility will have many advantages by which the most important is to have a communication point between the information producers and information consumers. Figure 8.2 shows the initial metadata browser components to be implemented based on the prototype design given in appendix A.

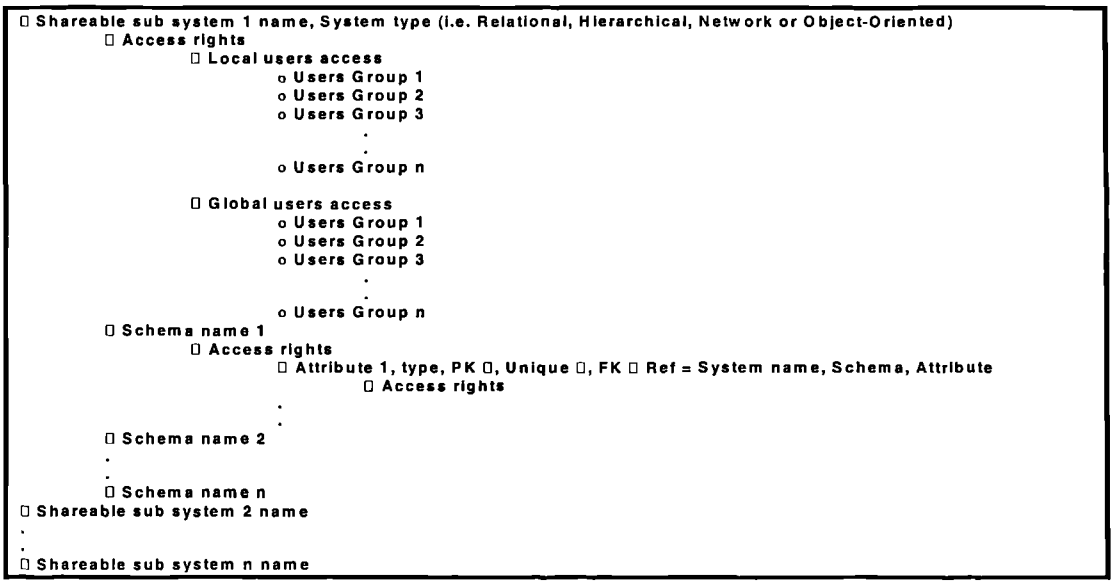


Figure 8.2

The overall goal of the work in the database interoperation is to demonstrate a new, modular and dynamic framework for the knowledge handling from the information producers and consumers side in the distributed, autonomous and heterogeneous databases that is necessary for the interoperation process. The framework will consist of the necessary parameters that should be known to the interoperating information sources and how the local database applications will be able to interoperate with other heterogeneous information sources via a proposed layer. A key aspect of making the knowledge about the interoperating databases available to the information consumers involves assigning a unified metadata handling protocol, which is imperative for developing methods that can perform tasks with as little human intervention as possible. Such interconnected information sources supporting the sharing of data may be called cooperating information sources by which the design, construction, use and evolution of such system within the above paradigm will require sophisticated technologies from many different areas of computer science.

The cooperation between the heterogeneous distributed data sources can take two distinct paths. The first, which forms the major demand from the interoperation of databases, belongs

to the normal information gathering needs. This means the requesting user has an information pool by which he needs to share it and cooperate with other people owning similar information so that he can get better statistics for better business decisions. Such type of requirement increases in areas like the medical statistics. The second path is considered when an information source allows updates from the global either by anybody or by specific people. This type of operation is very much similar in nature to some existing Internet HTML services. In reality it is an advertisement about existence of information pool and cooperation between information sources of possibly different schema types. The difference in the normal HTML services is they don't know about the other information sources available around. While in the case of the IE the information source will be open in a sense to give the information services to the information consumers in a way by merging their information pools with other information owner pools having similar information. In the case of the IE the distributed information consumers will be informed about the available information they can access from the global information producers sources. The information consumers will be able to use their own data entry screens of their database applications rather than having to use a pre-prepared data entry screen, which means they will not be able, to merge their information pools with the other related information pools. Furthermore, when adding new record or editing an existing record is allowed then this type of operation should apply the stipulated constrains assigned by the information owner.

8.2 Overview of the IE

The most important feature is that the approach of databases cooperation ensures no effect on the other normal services offered by the database management systems. Among those services are: schema translation management, programming language translation management, semantic inconsistency management, and other aspects related to the operating system and the communication protocol layers which are considered as outside the database research realm.

All these issues make a strong demand towards having a new dynamic mechanism for the cooperation of the heterogeneous distributed data sources. This issue needs to be looked at in a new way. Although there are currently a number of suggested static definitions for the cooperating data sources, no one is considered as the ideal solution for the rapidly increasing number of cooperating heterogeneous distributed data sources. The Disco project [Tomasic96] is one example of the static definition of the necessary databases interoperation parameters and which this view is considered as a dynamic implementation of parts of the Disco.

At this stage it is important to provide an infrastructure capable of linking the heterogeneous distributed data sources in an incremental manner and for only those related portions of the data sources. This infrastructure is also responsible for sending queries to all the cooperating sources asking for certain information of interest. Also, this infrastructure should deal with all the users in the globe having access to its local data sources. The main contribution of this work is an infrastructure for an interoperation atmosphere serving the heterogeneous distributed databases and having a behavior similar to the Open Shortest Path First OSPF [OSPF99] routing mechanism.

OSPF is a routing protocol used within larger autonomous networks in preference to the *Routing Information Protocol* RIP, an older routing protocol that is installed in many of today's corporate networks. Like RIP OSPF is designated by the Internet Engineering Task Force IETF as one of several Interior Gateway Protocols IGPs.

Using OSPF, hosts that obtain a change to a routing table or detect a change in the network immediately multicast information to all other hosts in the network so that all will have the same routing table information. Unlike the RIP in which the entire routing table is sent, the host using OSPF sends only the part that has changed. With RIP, the routing table is sent to a neighbor host every 30 seconds. OSPF multicasts the updated information only when a change takes place.

A middle engine has been proposed, which can act as an add-on facility on current Internet browsers, responsible for binding only these heterogeneous distributed data sources of interest. This middle engine will not require the set of related information providing applications to agree on one global view. This approach is based on information availability and information demand. It will also depend on the information advertisement technique for the purpose of advertising an available information source. The infrastructure will also form the foundation for all the supporting areas in the distributed databases to take place towards the supporting of heterogeneity. The proposed steps to be taken for registering the metadata, advertising about the shareable parts of the local information sources and delivering the information parts to the distributed information consumers were presented in Figure 2.1.

Figure 8.3 illustrates the connectivity between the underlying system components and the functional relationship in the IE in a detailed and concrete fashion. The user can have his/her own domain where he can create his/her own data sources access profiles by which he can access them in his/her own applications. As another option he/she can rely on the profiles created by his local DBA where the DBA in any site is the link between the local information consumers and the global information producers.

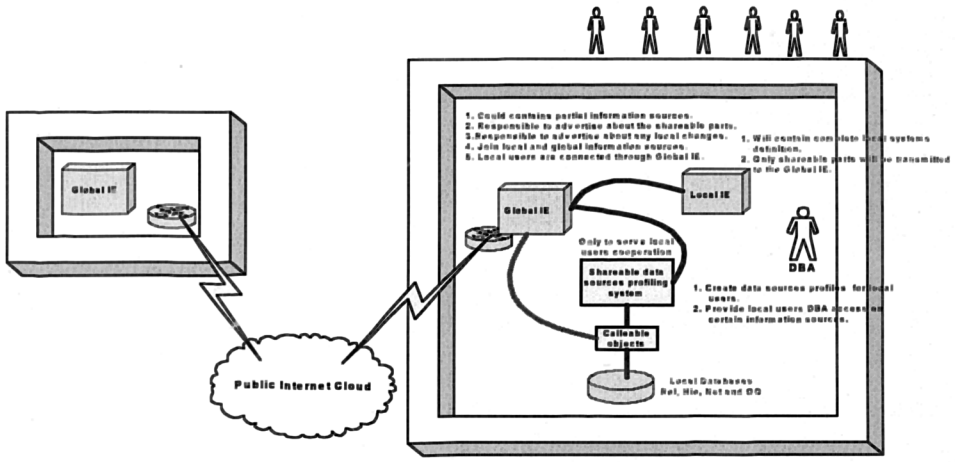


Figure 8.3

8.2.1 Querying for information

Both local users and the local DBA for gaining certain information can submit queries to the local IE. The local data dictionary is the main part where all the submitted local queries are processed. When a query is submitted, the user is given the available information space that answers the query and the existing data sources access profiles by which he can use in his/her application or query code. There are some other cases where some information owners may advertise about the availability of certain information in their sites without giving access to any global information consumer until a request is received asking for such access. Information owners may give access to global information consumers based on, for example, the further mutual benefits they may gain or for any other security purposes.

8.2.2 Exchanging information between participating IEs

The local IE information system contains a definition of a complete local systems. The shareable parts of the local systems are defined as routes and only these parts are transmitted to the global IE information system. In each of the cooperating sites the global IE information system link the local shareable subsystems with the other global cooperating databases. Any alteration in the global shareable space is mainly done locally in the local IE information system and then migrated to the global IE information system. This stipulation is to maximize the reliability of the global IE information system and to make sure that no long intervention will occur in the cooperation responsible subsystems. The initial partials local IE

information system UML based classes are shown in Figure A.1 (Appendix A) and Figure A.3 shows the global IE information system.

The shareable data sources profiling system is where database profiles are created and accessed by the local application programs. Basically it links information about data sources from both the local and global IE information systems.

Although, the design of the IE in its preliminary stages, this will involve assigning an information handling strategy between the cooperating data sources in which the exchange of the updates for the necessary metadata between the information sources can be done. The strategy should guaranty the maximum reliability and minimum intervention to the interoperation services.

8.3 Description of Participating Components in the IE

The actual building infrastructure of the IE is assumed to be the Internet by which nobody owns the backbone. Figure 8.3 displays the overall architecture of the different IE components and how they are linked together. It can be seen that the main components are the Interoperation Engine layer; the heterogeneous databases repositories and the metadata proxy server.

Subsequently the *IE* local knowledge consist of three interrelated components of which each plays an important rule in the success of the interoperation mission of the distributed heterogeneous databases. The three interrelated components are (1) *users and shared systems profiling*, (2) *heterogeneous schema management*, and (3) *cooperating schemas profiling*. Those are explained graphically in Figure A.1.

8.3.1 User and shared systems profiling

The IE system supports two type of users. These are the ordinary information consumer and the IE administrator. The one who will mostly benefit from the IE is the ordinary information consumer. The administrator is the person who will create users profiles, systems profiles and access profiles. User profiles are the grouping of the local and global users according to local site policies. Systems profiles are the same as user profiles but on the local and global systems. Access profiles are the link between a user profile and systems profiles. The following example explains the profiling technique in the IE system including user management. Figure 8.4 shows the relationship between the class diagrams for both the local and global profiling management subsystems. For more details regarding the participating components, see the figures of Appendix A.

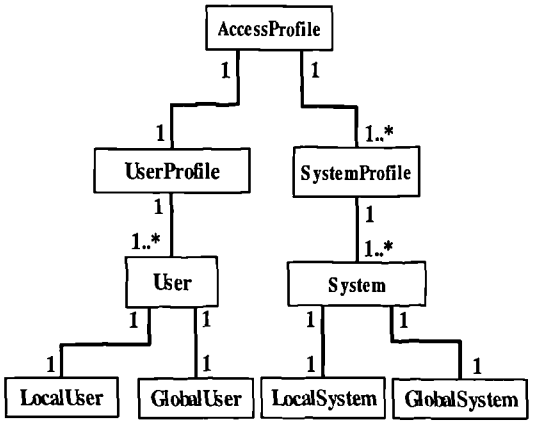


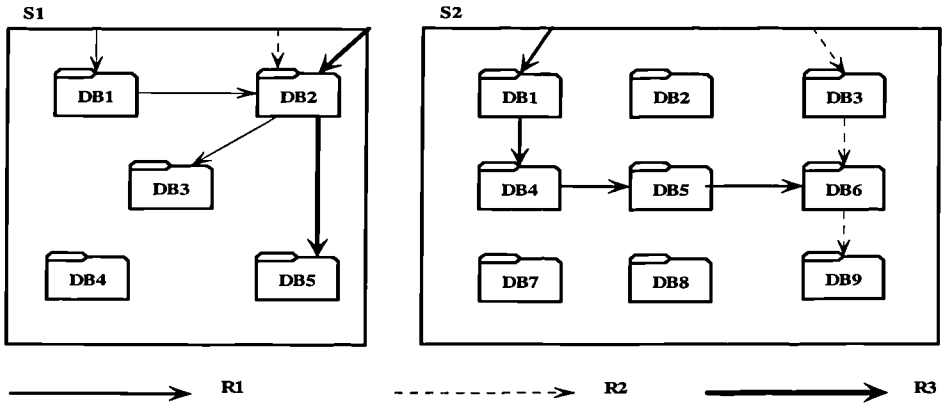
Figure 8.4

Initially users and user profiles are defined locally in the local profiling management subsystem for all users of a single site. Only those who will share global data sources are listed in the global profiling management subsystem. The local administrator in each site is able to access the global profiling management subsystem to assign global users to local

system profiles. Also, he/she who creates the cooperating schemas profiles to be used by the database application programmers, which was thoroughly discussed in Section 4.3. The breakdown shown in Figure 8.4 the IE is intending to send only the shareable parts of the information sources to the global information consumers. Access permissions of the global users on the local information sources will be checked two times. The first in the query originator IE and the second in the target IE and the synchronization and accuracy of the applied information will be checked prior to the query submission. Queries or requests are only passes if and only if information in both sites the same.

Having dedicated information for the local site and other for global sites will increase security and reliability of information. Local site will have total information about the existing information systems. However, only the part of the information sources supposes to be shared with the global users are copied in the global profiling management subsystem.

As indicated earlier, profiling simply means grouping. Profiling is one of the proposed facilities to be provided by the IE. The main purpose of profiling is to make the work more controllable, manageable and traceable. The IE mainly supports three types of profiling techniques: user profiles, cooperating system profiles, and access profiles. User profiles are simply grouping users into groups according to different policies such as they have similar access on certain systems or maybe according to user's original site. Cooperating systems profiles are the parts users will get access on them. Each site may have many systems by which each system has number of subsystems forming the possible accesses in a local database system. Each group is known as a route in the IE. Figure 8.5 explains how single site schemas will looks like.



Local IE systems and the routes supported by each system
Figure 8.5

As shown in Figure 8.6 the local site mainly supports two systems (will be called as S_n for system and n for the system number) where each system has a number of routes which can be accessed by the global users R_n. Furthermore, DB1 may consists of a number of tables (t_1, t_2, t_3, ..., t_n). Tables also will consist of number of attributes (a_1, a_2, a_3, ..., a_n). System 1 as shown supports three routes.

S_1: (R_1, R_2, R_3)
S_2: (R_1, R_2).

The routes in each of the two systems are defined as follows:

$S_1.R_1$: (DB1, DB2, DB3)
$S_1.R_2$: (DB2)
$S_1.R_3$: (DB2, DB5)

$S_2.R_1$: (DB1, DB4, DB5, DB6)
$S_2.R_2$: (DB3, DB6, DB9).

On the other hand the local site have got four users profiles as follows:

UP_1: (U_1, U_2, U_3, U_4)
UP_2: (U_1, U3, U_7, U_9)
UP_3: (U_{10}, U_{11}, U_{12})
UP_4: (U_{50}, U_{70}, U_{90})

Also, the local site has decided to establish the following system profiles (SPs) on the two defined systems from the defined routes according to private information policies.

$S_1.SP_1$: ($S_1.R_1$, $S_1.R_2$)
$S_1.SP_2$: ($S_1.R_1$, $S_1.R_3$)
$S_1.SP_3$: ($S_1.R_2$, $S_1.R_3$)

$S_2.SP_1$: ($S_2.R_1$)
$S_2.SP_2$: ($S_2.R_1$, $S_2.R_2$)

Once the system profiles and user profiles are defined then connecting users with systems becomes an easy task. The following is the connection of user profiles to system profiles, which is called access profile AP_n.

AP_1: UP_1 can access $S_1.SP_1$, $S_1.SP_2$
AP_2: UP_2 can access $S_1.SP_1$, $S_2.SP_2$

Now, any additional system profile which will be added later to the user profile are added to the list related to that user profile.

Until now the consideration is that users are permitted access on the full schemas defined as DB_N. In some cases the database owner may need to specify the grants a step forward by giving access to certain attributes within some schemas. In this case he/she should specify routes contains certain schemas and attributes. These will be defined in the global shareable subsystem in each local IE and fired to the global IE side when required. It is clear from Figure 8.5, that S_1 mainly supports three different routs, which appears in the global IE as different systems entities. If $S_1.R_1$: (DB1, DB2, DB3) has recalled with the assumption that DB1 has three attributes (A_1, A_2 and A_3), DB2 has four attributes (A_1, A_2, A_3 and A_4), and DB3 has again four attributes. Here still more than one route on the same three databases showing each time different attributes to be supported by each route could be defined. In this case the subsequent details, which will appear in the route definition, are the attributes supported by each route. The routes in each of the two routes of S_1 can be defined as follows:

$S_1.R_1$: (DB1[A_3], DB2[A_3, A_4], DB3[A_3, A_4])
$S_1.R_4$: (DB1[A_2], DB2[A_2,A_3, A_4], DB3[A_2, A_4])

8.3.2 Heterogeneous schema management

As indicated earlier, the local administrator is the one who makes the data sources ready for application programmers by creating the required application profiles and attaching them to the user profiles as access profiles. In this case application programmers access system profiles in their applications rather than directly accessing the original data source. This step has got many facilities of which the application programmers does not need to change in the application source code when new data sources added into the cooperation process. The following few paragraphs are explaining in an example how the distributed heterogeneous information sources are managed. Also, explain how application programmers should deal with the profiles of the IE in their applications.

The cooperating data sources are either contains similar or dissimilar information. If they contain similar information then the possibility operations between them is unification, intersection or difference between the records. In the case if they dissimilar the only possible operation between them is relationship through some common field in both that can be considered as a primary and foreign keys.

As indicated earlier, the cooperating information sources will be managed through the global schema profiling subsystems. It will be responsible to apply the three operations in between the master information sources with the other distributed information sources. The three operations are unification, intersection and difference. Figure 8.6 shows preliminary contents of the *global schema profile.*

```
Profile name: ABC system schemas profile

    Local schema name: DB1
    Operation: Union
    Involved schemas: S1, S3, S4, S7, S8, S9, S20, S34, S99

    Local schema name: DB2
    Operation: Relationship
    Involved schemas: S199, S333, S114, S237, S548

    Local schema name: DB3
    Operation: Intersection
    Involved schemas: S101, S301, S401, S701, S801, S902
```

Figure 8.6

Figure 8.6 shows if the case is a relationship as the operation applied on DB2, then here are different possibilities. The first, is when the relationship applied to a single remote schema then this is considered as a normal case. The second is the abnormal, i.e. when the relationship is with multiple remote schemas then it requires the application of an operation to the cooperating information sources to get them as a single information source prior to setting up the relationship linkage.

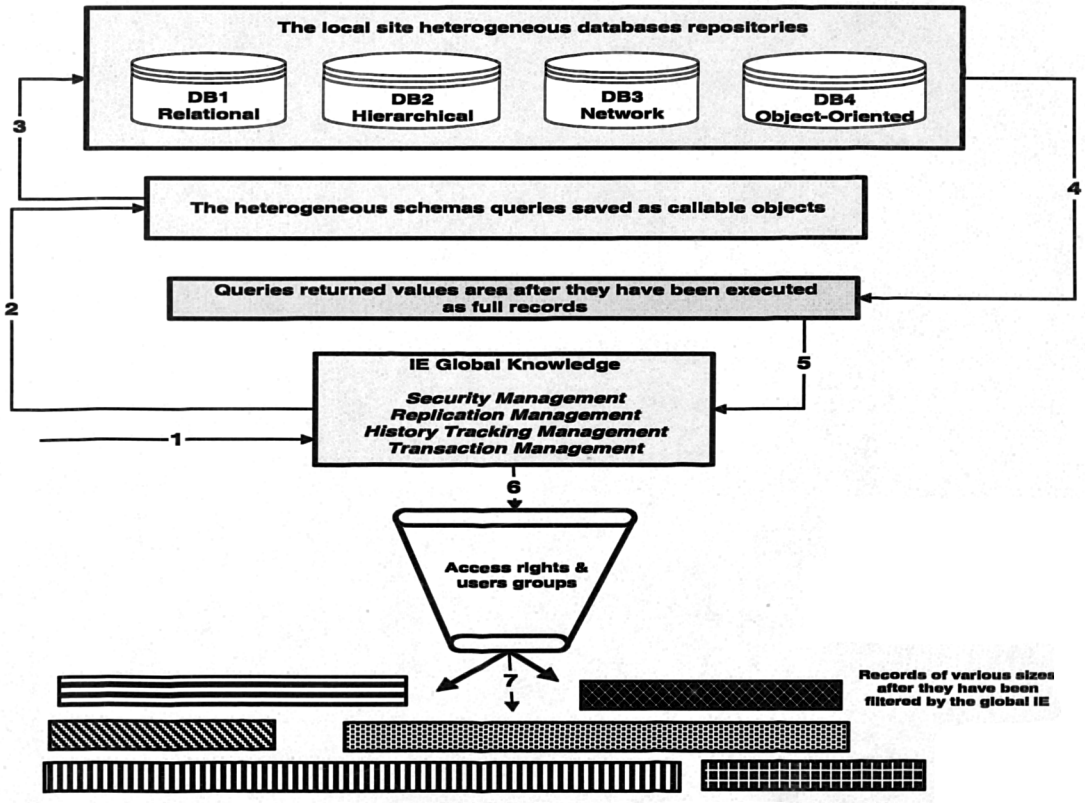


Figure 8.7

The proposal is the application program not directly reading from the remote data sources. Rather, it is done through the *global schemas profile* facilities provided by the local IE, which mainly liaises with all the related global schemas in a manageable, expandable, adjustable and dynamic manner. From the operating systems side there will be some extended

requirements to make the cooperating information sources linked together, as well as, synchronize with each other. This task is purely related to the operating systems since they will be responsible to share memory and open files between the operating system processes, protect pages of memory which has the cooperating information sources records and linking the distributed cooperating information sources memory pages together considering the sorting constraints.

The main advantage of this approach is the creation of the cooperation atmosphere between the different information sources which can be achieved by an incremental manner. Also, this cooperation process will not involve changing in the already compiled and tested source. The user will only change in the profile related to the system using the IE provided facilities.

According to the IE stated design, the information producers will not be able to interrupt any running query against their information sources. The changes in the permitted information space will be carried out in two stages. The first stage involves the changes on the permitted information space prepared through the local IE. Once the changes finish it will be migrated to the global IE side and the old information space will be replaced with the new one. Figure 8.7 illustrates how the IE deals with the queries and where the filtration against the access rights are applied.

This technique also has another advantage where the global part of the system will not be interrupted, as well as, modifications in the available information source for the global can be accomplished in the local site and then fired to the global. Changes will take place and the necessary parameters in the global IE services will be modified.

8.3.3 Cooperating Schemas Profiling

As explained earlier, application programmers depend on accessing data sources profiles rather than actual databases. The data sources should be available to them so that they can call them from within their application. Databases are called one at a time from within any DBMS programming language. The database administrator is the one who normally builds the data sources profiles that may consist of more than two information sources. He is the one responsible to give the information consumers the different reports about the provided information source they gain from the global information owners. The plan is also to let the individual users having access on global information sources to create there own cooperating schemas profiles.

At this stage the data sources profiles are available be called from within the application program. Assume that each of the three databases has a number of attributes, and the first database DB1 that the example is about having three attributes (A1, A2 and A3). As a normal case the three attributes may be accessed by their names in the application program. In the case if there are other cooperating databases with DB1 than the access to the attributes is through the use of the data sources profile. Assume the local system is able to cooperate with other three global systems having equivalent data as the local system. Those are S2, S3 and S4 respectively. Assume again S2 database 1 has four attributes (A1, A2, A3 and A4), S3 database 1 has again four attributes (A1, A2, A3 and A4) and S4 database 1 has five attributes (A1, A2, A3, A4 and A5). The definition step of the equivalent global attributes to the local attribute is given by

Local A_1 = $S_2.DB_1.A_1$, $S_3.DB_1.A_2$, $S_4.DB_1.A_4$
Local A_2 = $S_2.DB_1.A_2$, $S_3.DB_1.A_4$, $S_4.DB_1.A_1$
Local A_3 = $S_2.DB_1.A_4$, $S_3.DB_1.A_3$, $S_4.DB_1.A_3$

There are two possibilities when linking the cooperating data sources with the local application. The first is when local application accesses a local database as in the above example where local database cooperates with global systems S2, S3 and S4. The second is when the local application doesn't have any local database and will directly access remote global databases where in this case the local attributes will be considered as the local application parameters.

8.4 Metadata Handling Protocol in the Distributed IEs

The Extensible Markup Language XML will have much input to the interoperation process. In that, the XML has added a powerful transport mechanism to the rapid database interoperation requirements. Sense the different information handling mechanism between the different interoperating information sources can use this language as a method for putting structured data on a text stream. The transmission of information should depend on the policies that are assigned by the proposed interoperation engine IE when applying the different database operations.

Handling protocol of exchanging the metadata within the cooperating database systems is considered as a strategy task by which all the cooperating sites have to agree on certain handling methodology. This section does not form any comparison to any international standard rather than considering a methodology that fits the different requirements of the heterogeneous database interoperability. It is understood that the functional programming languages may contribute in the design of such protocol, but at the time being this is beyond the scope of this chapter. For the sake of this chapter, a protocol which satisfies the initial requirements and defined by the IE prototype has been proposed. The protocol, which is XML based, is assumed to deal with the shareable database systems metadata and all the related information for the purpose of exchanging this information for all the cooperating IE sites. Since each schema type has its own definition procedure with different constraint assignments it would be difficult to provide a unified protocol to handle all the schemas. As a first attempt it is planned to tackle the general protocol shape in this section. A further step is to extend this protocol i.e. to make it unified as possible in order to handle any schema type requirements. Figure 8.8 explains the basic required information to be processed by the proposed protocol.

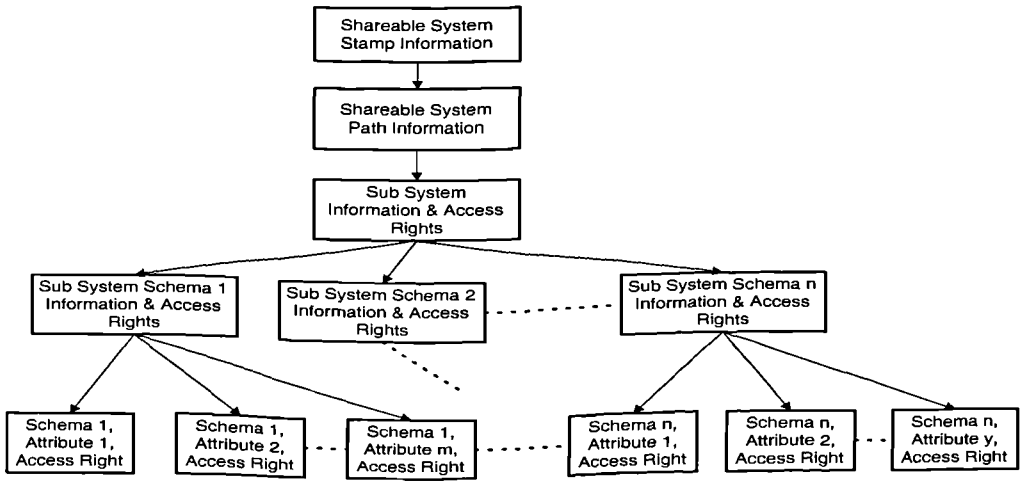


Figure 8.8

Based on the above figure, if the relational schema shareable system defined as displayed in Figure A.2, the global IE is considered, then a definition for the different components to cover the whole schema design is needed.

In this chapter, architecture has been presented by which a distributed and heterogeneous information sources can cooperate together in an interoperation atmosphere to share information taking into consideration the current advances in the Internet facilities. More specifically, a mechanism that enables information producers and consumers to have a common pool by which they know each other and the information space provided for each has been described. This mechanism accomplishes the goals without any effect on the autonomy of the participating information sources. The plan also is to conduct more research in refining this architecture. Some parts of the prototype has been implemented and working to implement the prototype as a whole to validate the design ideas presented in this chapter.

Several issues for sharing information in a cooperative manner across autonomous distributed heterogeneous information sources have been addressed. It is well known that the more there is sharing, the less autonomous databases are. For instance, the use of schema integration increases data sharing dramatically while bringing database autonomy to nonexistence. In contrast the above-described prototype designed to increase cooperation between the distributed heterogeneous databases without dimensioning database autonomy. This approach provides a mechanism that enables database application users to be informed about the available information space they can gain from the globe information producers. It also enables them to share information with other information holders in a transparent, expandable and autonomous manner. The design of this prototype is established in order to contribute with metadata capable web proxy servers so that they become heterogeneous databases accessible web browsers.

Chapter 9

The IE Supporting Services

The most important operation in the cooperation process is the process of the integration between the heterogeneous data sources. The integration is simply the hashing technique between the different heterogeneous data sources elements. Further, the migration of the data records takes place between the data sources. The IE proposal has defined such framework, as well as, defined some other assistant services for facilitating the interoperation process. On the other hand, there are some other assistant services facilitate the cooperation process such as the syntactic and semantic data dictionary resolvers, history tracking tools, data replication management and many other supporting services. The IE design is capable of incorporating within its processes many facilities that could simplify the cooperation process. In the coming sections illustration to the design tips that could take place in conjunction with the IE design will be discussed.

Regardless of the schema type the database management uses, there is also requirement for assistance services that can improve things such as standardization of the application, standardization of the naming conventions used by the database schemas, tracking facilities and other service availability related tools. Achieving such goals in an environment where different database schemas are interoperating is an important requirement where the gains behind are major. The tree like design of the IE does simplify the existence of such services. In the following sections three different services have been discussed: syntactic/semantic data dictionary, history tracking and the replication services for the heterogeneous distributed databases. The purpose of such discussion is to highlight the IE design capabilities that could be invested in the existence of such important assisting services.

9.1 The IE Syntactic/Semantic Data Dictionary

9.1.1 The IE Syntactic/Semantic Data Dictionary Design

It was reported in Chapter 8, the backbone of the IE prototype proposal is assumed to be the Internet by which nobody owns it. Also, the assumption is this part supposed to be active throughout the lifetime of any local interoperation process. The process will be responsible for clarifying the meaning of a data definition to the global users during the interoperation process between the distributed heterogeneous data sources. This will help the different users in matching between the different naming on the different distributed sites. This subsystem should co-operate in solving the problem of semantic inconsistency [Stonebraker94], and make users queries to reflect the actual needs as much as possible. Semantic inconsistency across heterogeneous information systems is a much more complex technical problem with no general solution yet devised, and it is still an open research challenge. Also, by the assistance of the dictionary it should come up with techniques that will identify interesting patterns of information and thus help the user to be aware of potential sources of interest. Users should be able to issue queries by the assistance of the dictionary and get the proper answers without knowing which remote sites are involved in answering the query. The sites involved in answering the query could be found by the assistance of a link tracking subsystem responsible to start the query by taking the links from the data dictionary subsystem information knowledge, which is capable of keeping track of the different processes in the interoperation. The extended plan of the data dictionary of this sort is to provide the users with the necessary links about where to find certain information. The data dictionary is capable of including the link to the site that provides certain information in case the information producer is interested to provide the information to the consumers directly and without any permission restrictions.

As an extended proposal, this part of the prototype is also responsible for the advertisement process about the availability of certain information or the registration of the areas where information producers and consumers are interested in and where this information can be found. Information of interest can be registered at this stage and stored at a common pool by which it is reachable by the information consumers. The advertisement can do two different tasks. On the first hand, the information producer who is at the same time an information consumer can advertise about the information he owns. On the other had, he/she can also ask the other information producers about certain information type. Through the data dictionary usage it would be possible to link the information producers with each other and to glow the similar areas of interests with each other sense the data dictionary will have the different areas of interests and their related sites. The initial plan is to make this part of the process to deal with the international e-mail services and to update the areas of interest whenever new area appears. Figure 9.1 is the UML based classes and the relationship between them. Both the SystemStamp and the SystemPath classes are part of the data dictionary implementation.

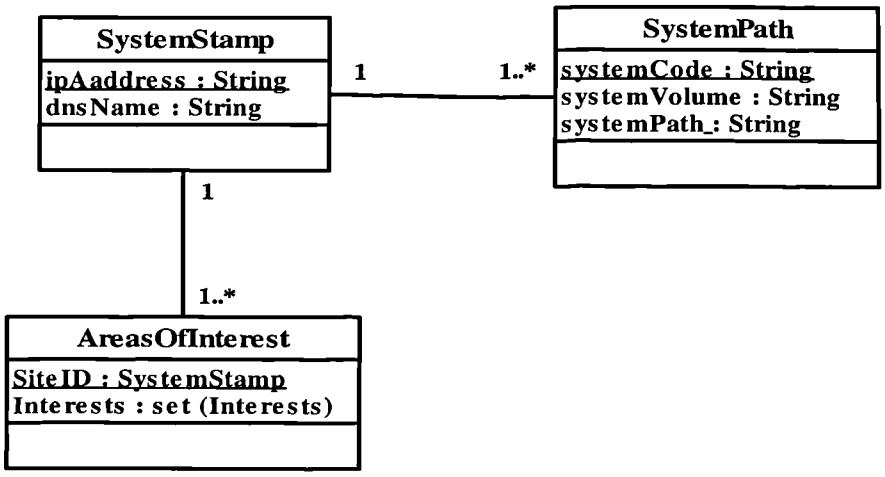


Figure 9.1

9.1.2 The IE Syntactic/Semantic Data Dictionary building blocks

The semantic data dictionary SDD is built from three components: servers, query clients, and the SDD meta information. The SDD server main duty is to store data dictionary information related to their client users. As a secondary task for the SDD servers is make connection with other SDD servers to resolve clients' queries. The query clients are the members of the cooperation. They can send queries to their SDD servers. As the IE indicated, most of the queries will be handled transparently while the user deals with the different database components. The only static queries are the ones done by certain users searching for specific information. This type of queries will be broadcasted to every global IE registered by the global system stamp class. Since the SDD is distributed, it is also necessary to store where a domain can be found. This can be done with the help of the global system stamp class in the local IE system.

The data dictionary knowledge base is a single class consists of attributes such as the code, the database element name, its type (system, table, or attribute), the description of the database element, and a set consisting of the likely links to the information sources where the information has been mentioned. As an extended facility it has a list of the system stamps they appears into them so that the query can be done across the sites.

The content of each of the data dictionaries must be maintained, which is the responsibility of each of the local IEs. For this reason each IE host must manage all the database components and provide the semantic meaning for them.

9.1.3 The IE Syntactic/Semantic Data Dictionary architecture

Without clear architecture for the semantic data dictionary the semantic conflicts in the cooperation between the distributed heterogeneous databases cannot be effectively resolved. Possible architecture is shown in Figure 9.2 for the SDD architecture where a distinction is made between the backbone SDD and the LAN SDD. The backbone SDD has information about the whole Intranet. Within the backbone a division is made into master SDD server and regional SDD servers. The master backbone SDD server passes data into the regional SDD servers. Each IE SDD has information about only the related cooperation information of its own. The local IE SDD asks the regional SDD for the information that is not known locally. In this case, the reliability increases as the backbone SDD is geographically dispersed.

Query clients are considered as automatic resolvers in the above scenario. They first approach the local LAN SDD. f they did not found what they want to know then they can fallback to the backbone SDD. This arrangement minimizes the usage of the bandwidth of the wide area network and the time of the process until it gets done. Also, the local SDD will always hold the necessary information related to the local cooperation requirements. With this setup the required information about any database component will be known as quickly as possible. Furthermore, this set up gives extra robustness in two ways. Getting the necessary information about any database component is applicable either if the local SDD is down or the backbone SDD is down provided that the backbone SDD servers are reachable.

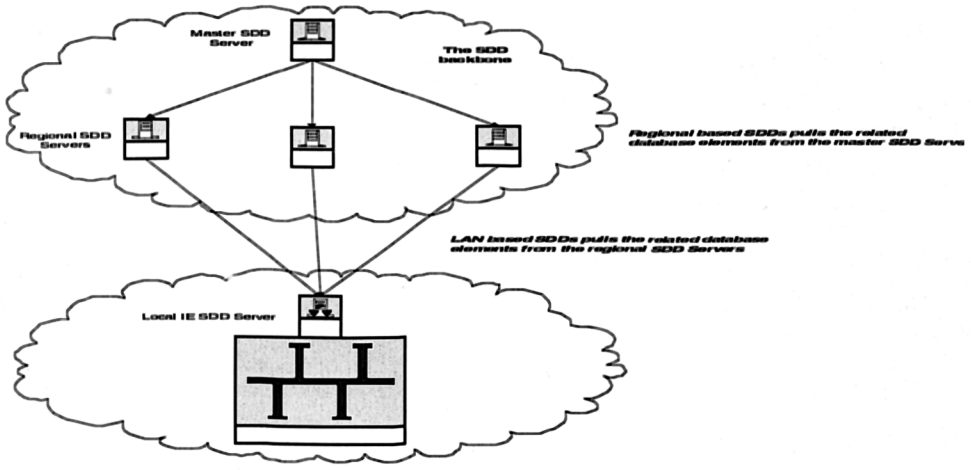


Figure 9.2

To make the optimal use of the SDD services, a naming convention is vital. A naming convention defines the architecture of the content. It describes the structure and roles for the database components naming that are in use within a single IE domain. A practical naming convention ensures that names need to change as little as possible, and that they have a logical structure. At the same time, the naming convention must also have both a technical and a human dimension. The technical dimension gives the relationship between the different database application components. The human dimension is concerned with ensuring that the end users and administrators can apply the convention in practice.

9.1.4 The process

The process of assigning the SDD clarification starts at the metadata definition phase of the database. The interaction starts with the local IE SDD server that will provide all the related syntactic/semantic clarifications. While the information source owner defines his/her own database elements he will be in a position to clarify the meaning of each of the database elements. This clarification could either be selected from the local IE SDD knowledge base or it could be new element added to the local IE SDD repository. To prepare the new added databases to be ready for the global user, an advertisement about this new information need to be sent to the related information consumers. According to the individual application owners' business rules the clarification to the database element will be broadcasted to either selected partners or to the global where the updates will be sent to all the global master SDD servers. The updates will be broadcasted to the selected partners if they are predefined to the

local IE prior to the syntactic/semantic clarification process. Otherwise, the updating step could be deferred until a business requirement occurs by the data owner and a late update can be taken as a later process.

So, as planned for the IE SDD, the final information will form a standard naming conventions, as well as, it will clarify the conflicts that maybe caused by different users using different languages and different business requirements.

9.2 The IE History Tracking Manager

9.2.1 The IE History Tracking Manager Design

In case the process fails the History Tracking Manager [Lee97, Barbra91] has the capability to resubmits the process again from the point it fails. This part deals only with the actual query submitted to the IE. According to a predefined time a collection of the reached record values will be stored by the IE. The resubmission of the process, which completes a suspended process and the glow operation of the finished part with the resubmitted part, is the responsibility of the Transaction Processing Manager that is initially not included in this thesis.

History tracking management is one of the most important services in the distributed heterogeneous database environment, as well as, the other distributed services providing queries on remote databases and are subject to suspension. The most important benefit of such facility is the restart of the suspended processes from where they have been stopped. Although, this task should be handled by the DBMS where the query runs and the Operating System where the DBMS works under. Here a highlight to the facilities that could be gained from the IE design and how to make the maximum benefits by including such services in the distributed heterogeneous database interoperation process has been given.

The history tracking can only be applied to processes i.e. queries which are using sorting facilities such as the indexing technique used in the relational and object-oriented databases. The main problem causes not being able to use the full functionality of the history tracking in the legacy hierarchical and network databases are because they are not table-based databases. Rather, they are database-based databases. Also, the redundancy in both the hierarchical and network databases makes the tracking management facility a difficult task for such databases. There is no other way than tracing the databases serially in the hierarchical and network based legacy database. On the other hand, in the relational and object-oriented databases it will be possible to go to the point where the query has suspended by locating the place of the suspension in the resubmitted query. In such databases, making use of the indexes by using the binary chop algorithm could be done [Islam97].

The analysis of the binary chop algorithm is a little more complex but can proceed in the same way by considering the best case, worst case and average case situations. The best-case situation in this circumstance would be for the item to be located in the middle of the list, in which case it will be found on the first iteration. The worst case situation, where the item being sought is not in the list, is a little more complex to determine and can be approached by considering lists of different lengths. For a list of one element it will take single iteration of the main loop for it to be determined that the single item in the list has not been sought. For a structure with two elements two iterations would be required. The first iteration will decide that the item been considered has not been sought and the second iteration will be restricted to a sub list contains the remaining element. Figure 9.3 shows the summary behavior table of the binary chop algorithm.

Order analysis for the binary chop search algorithm

Best case	1
Worst case.	$\log_2 n$
Average case if item is not in the list.	$\log_2 n$
Average case if item is in the list.	$\sim\log_2 n$
Average case.	$Pf(\sim\log_2 n) + Pm(\log_2 n)$

Figure 9.3

This subsystem therefore forms a knowledge base and will be responsible for keeping track of the status of the operations taken by the local site base on certain defined time interval. It will also be responsible for continuing the suspended operations handled by the transaction manager. Also, this will be capable of giving the history about the sites where a certain command has been applied to them, because these types of operations should be handled in a transparent manner to the users. If certain critical operation fails this subsystem should be capable of exactly mentioning where the failure has been occurred and forward it to the transaction manager, which should be responsible for the continuation of uncompleted processes.

In cases where the legacy database schema is hierarchical based or network based, the resubmission of the suspended operations will not gain the maximum benefits of the history tracking management if restarted from the point where they have been stopped. The only gained benefit is the time saving of the already delivered part of the original submitted query. This because the tracing operation in both database types for reaching the point where the interruption occurs is done purely serially in the present work.

9.2.2 The IE History Tracking Manager building blocks

The history tracking manager requires dealing with three interrelated components: time, query and the resubmission of the adjusted query in case of failure. The time is considered as the interval between a query status at certain time and its status at the next registration time. In this work, a status means that the record that has been reached at the predefined time interval. The purpose of defining the time in a static manner is to retain the response time, since the history tracking operation is considered as a burden process on the running systems' response time. The resubmitted query in case, of non-completed query which maybe caused by any of the failure reasons has to be modified according to the point where is was reached in the original query. In this case the IE where the failure occurs for a certain query will send the failed query identification to the IE that sends the query by which the sender will only send the resubmission again. When the remote IE receive the resubmission request it will resubmit the query based on the information it holds about where the original query reaches.

9.2.3 The IE History Tracking Manager architecture

The building architecture of the history tracking manager is mainly depicted from the IE architecture. Since, it is considered as a complementary component dependent on the IE building components. Figure 9.4 displays the steps which form the architecture of the IE history tracking manager.

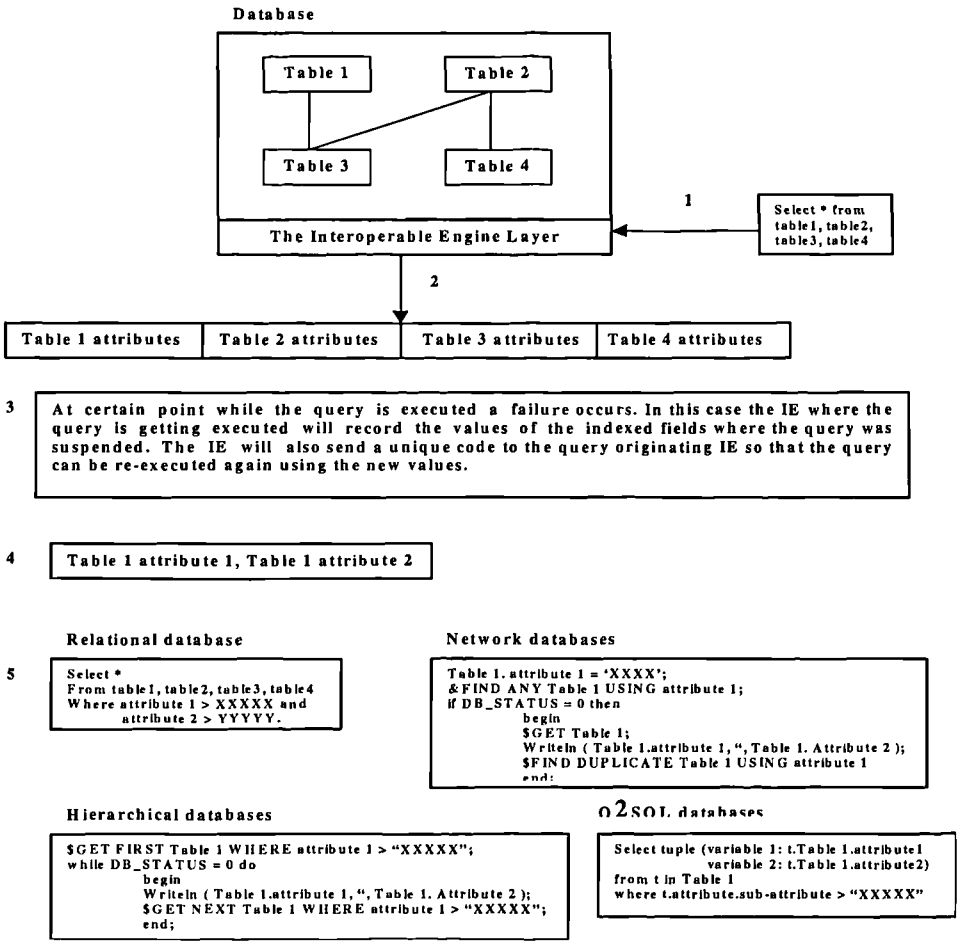


Figure 9.4

It is obvious from Figure 9.4 that the resubmission process of the stopped queries will be submitted to the IEs in the form of the original query modified to restart from where it stops. When the stopped query code comes to the IE it will modify the predefined query belongs to the original query fired by the information consumer and through the IE layer it will start to hand over the missing part of the information which is the continuation of the original query. The operation is, of course, done on the universal view that has been created on the information's owner side. Here, the view is referred as a universal because it is simply flat non-normalized in the sense that could be understood by everyone. The view will automatically create all the required indexes from the knowledge it has about the information source. The advantage of accessing the information through the created view is that the view will define the necessary indexing facilities to access the data even if the indexing facility does not exist in the original databases system than may run legacy database. Also, sense the view could be created to run by and DBMS or even a spreadsheet, then any composite index can be created for the users accessing the view.

9.2.4 The process

As shown in Figure 9.4, there are five steps to record before a query is stopped. The first is when queries are fired against the IE. First the IE will record the first record information and set the time of the next information taking process that is step four. When the time interval reached the previous information will be overwritten with the new values. In case of failure the values generated by step four will be taken and the original query is modified by the new values and restarted again by the IE at that site.

9.3 The IE Replication Manager

9.3.1 The IE Replication Manager Design

One of the most important facilities in the database interoperation is to be able to replicate similar information from certain database schema type to different schema type. As the original IE design stated, the replication management should form a security advantage when forwarding specific requests to the replicated pools. Furthermore, replication is useful in improving the availability of data. The most extreme case is the replication of the whole database at each site in the distributed system; thus creating a fully replicated distributed database. This can improve availability remarkably because the system can continue to operate as long as at least one site is up. It also improves performance of retrieval for global queries, because the result of such query can be obtained from the nearest site; hence the query can be processed locally if that site has copy of the data. The disadvantage of the full replication is that it may slowdown the update operations drastically, since a single logical update must be performed on every copy of the database to keep the copies consistent.

This manager plays the role of backing up the original site repositories or part of them, as well as, re-forwarding some requests to the replicated data stores for purposes such as load balancing or security enhancements. In the IE the replication process could be identified to any part of the database regardless of its design. As stated earlier, the database elements are defined as tree like in the IE design, which simplifies the replication management process. Also, when data are extracted from the data stores they get de-normalized prior to sending them anywhere. This feature will give the flexibility to write the record to any schema type since the whole information will arrive to the other site as a single non-normalized record, which makes it easy to be inserted in any schema design.

The simultaneous multithreading technique would simplify the replication process. Multithreading is a technique that permits multiple independent threads to issue multiple instructions each cycle. It is the ability of an operating system to execute different parts of a program, called threads, simultaneously. The programmer must carefully design the programme in such a way that all the threads can run at the same time without interfering with each other.

Normally, the site where new records are added and updated will take the replication decision to which site it will be applied. In fact, the replication process will only start after successfully passing all the constraint types defined in the master database application. Also, operations such as updating and deletion will take place any time the master database gets changes. The required knowledge for such operation will be the address of the database where the replication will take place. Assuming that the local relational database has a replication constraint on records or even part of the records to another remote hierarchical database then the relational related records should first de-normalize before packetized and traverse to the remote hierarchical replica database.

Replication manager [Narasimhan97] is the one capable of liaising with other sites to get permission where a backup data store can be created for certain critical databases. This should play an important role when the main site is down by transferring all the process to the backup site, and when the fault is over it should be capable of restoring all the changes to the master database and convert the process back to the original site.

9.3.2 The IE Replication Manager building blocks

The replication management process requires the existence of components to be achieved easily. It requires the existence of the database component that is required to be replicated somewhere, information about the place that will take the replication which is considered as the target for such operation, the communication management with the targets and the mechanism the will drive the operation. Replication management is not an easy task because it is considered as the main issue behind keeping backups, which of course affect the system reliability and availability measures. Normally, the replication could either be considered as an instant process, which is most of the time costly or it can be applied according to a predefined time schedule in the case of the used system is not highly critical. In the IE implementation an

initiation of a solid building for a replication manager that is capable of taking both replication management philosophies has been initiated.

9.3.3 The IE Replication Manager architecture

In normal database processes the replication in databases can be done for complete databases in the sense the replica could either be used as a backup or for load balancing purposes. The replication management that comes with the IE prototype proposal is done in a way to provide a replication for up to the level of single attribute. Also, the replication could be done from one schema type to another different schema type. Such facility would probably increase the interoperability functionality supported by the IE. Initially, the IE replication management process can make use of the two-phase commit that is already supported by most of the existing communication protocols and operating systems. This process will change the idea of the replication that is considered as a dump backup for data to a more sophisticated replication process capable of handling the interoperability of the heterogeneous distribute databases requirements.

9.3.4 The process

The basic requirement for the replication that should exist prior to the replication process is to have equivalent pool to the one that need to be replicated. The IE will do such task when mutual acceptance between the two places exists. The plan is that through the IE it would be possible to define an equivalent repository regardless of the schema type in use. The IE will take over the replication management risk by liaising with the different knowledge bases to make sure that the replication is dune in the light of all the assigned constrains on the replicated information. As explained in Chapter 8, among the available information in the IE knowledge base is the different cooperating sites with the exact path that is necessary for the communication process between the different global cooperating data sources.

9.4 The IE Indexing Unifier Manager

9.4.1 The IE Indexing Unifier Manager Design

An index is a tree structure consisting of a combination of attribute values and physical storage address that allows direct access to a row in a table. Indexes can be classified according to their logical design or physical implementation. The logical classification groups indexes from an application perspective, while the physical classification is derived from the way the indexes are stored. The basic index could be a single column index that has only one column in the index key, or it could be a concatenated index that is created on multiple columns in a table.

In order to implement the IE a more sophisticated indexing technique to support the different scenarios that may happen while the cooperation process between the various types information sources takes place. The first proposed scenario which may happens between two disparate information sources assuming that both information sources are using a formula to generate the primary key related to a record object. The formula avoids the generation of similar keys in both information sources. In this case although the record objects will have different identifiers they cannot be sorted correctly unless excluding all the other parts than the sort key from the index identifier. In this regard the IE propose an implementation of a universal indexing creation and management algorithms. Those algorithms are to facilitate the interoperation and cooperation processes between the heterogeneous distributed information sources. Alternatively, as a second scenario, the index file is to contain additional pointer to the information source only in case the record is not belong to the local information source.

9.4.1 The IE Universal unified indexing architecture & Building Blocks

Basically the IE universal attribute identifier considers four parts accompanied with the attribute to make out the universal identification reference. Those are the system stamp class that contains information about the IP address and the location of the information sources, the system path, the sub system that attribute belongs to it, and the related table where this attribute is a member of it. Here, the plan is to sort the index file on the attribute object identifier so that in case an information source that consists of records from disparate

information sources they will be in order. The only obstacle about this technique is when records are accessed while the site is disconnected, which will give incomplete information. Figure 9.5 shows the different building blocks of the IE universal record, i.e. object, identifier

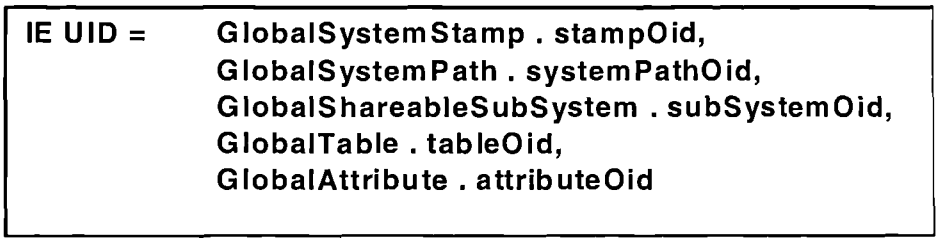


Figure 9.5

9.5 Benefits of the IE Supporting Services

All the supporting components are considered as complementary processes. Such components form the ideal and balanced interoperability atmosphere for the non balanced heterogeneous and distributed information sources willing to cooperate and exchange information. The syntactic/semantic data dictionary will provide the meanings of the different information sources components by which users will be able to use the links provided by this knowledge base without the need to link to the actual information source components which maybe not understandable. As explained earlier, the IE history tracking manager will trace the different information requests and will restart stooped requests from where they have been suspended. The replication manager is responsible for backup information sources to anywhere in the cooperation and for the load balancing when so many hits are met in any of the information providing sites. This has to be sited up by the information source owners. Furthermore, the IE indexing unifier manager is mainly responsible for reading records and keeping them under one unified indexing mechanism to guarantee fast retrieval of the record.

Chapter 10

The IE Unified Security Manager Design

Exchanging data across the world in a secure manner has become an important requirement among information owners that are willing to share and exchange their information sources with others. Information owners want to assure their autonomy on the information they own while they let others to share their data with them. In this sense, the various constraints assigned by the information source database management system, as well as, the application running over the information source has to be fully obeyed by both the local and global users accessing that information source. This chapter is devoted mainly to discusses the background of the distributed database security problem and the proposed solution.

In general, applications can either be Telnet based applications or a Client/Server based applications. The primary function of Telnet is to allow users to log into remote host systems. Telnet based applications are those applications using the Telnet programs to gain attachment to the application site where the operation is considered as if the link is local to the application site. There are many Telnet programmes where each has got its own specific characteristics and policies. Depending on the Telnet program policies, the best environment can be created to serve the connectivity to the remote application site. On the other hand, Client/Server applications are those intelligent applications called front-end systems interact with back-end server systems that provide services, such as database access, network management, and centralized file storage. This implies that the user has a computer with its own processing capability, which runs a program that can handle user interaction and data presentation. Thus, client-server computing replaces the centralized computing paradigm.

Telnet is the login and terminal emulation program for the TCP/IP networks such as the Internet [Siyan94]. It was originally developed for ARPANET but is now mainly used in the Internet sessions. Its primary function is to allow users to log into remote host systems. Originally, Telnet was a simple terminal programme that sends all user input to the remote host for processing. Newer versions perform more processing locally, thus providing better response and reducing the amount of information transferred over the link to the remote host.

Telnet is a client-server process in which the user invokes the Telnet application on the local system and sets up a link to a Telnet process running on a remote host. The user issues requests at the keyboard that are passed to the Telnet client running on his/her system. Telnet then transmits the requests to the Telnet server on the remote host. Through this process, users can initiate programs on the remote host and run those programs from their own system as if they where attached directly to the remote host. Most processes run on the remote host. It receives requests from the user's system and processes them in its workspace, thus reducing traffic over the wide area links.

For the Telnet users on the information sources applications there is no issue on constraint enforcement. This is because all the users will be treated equally by the accessed systems as if they are all local. From the research investigations on the telnet applications, such technique creates a bottleneck on the information source because basically the application-hosting server will apply all the processes.

In the client-server computing model, which is the focus of the IE security model, users work at intelligent computer called front-end systems and interact with back-end server systems that provide services, such as database access, network management, and centralized file storage. A computer network provides the communication platform on which many clients can interact with one or more servers at a time. The interaction between the users' front-end

application and the database program at the back-end server is called a client-server relationship. This implies that the user has a computer with its own processing capability, which runs a programme that can handle user interaction and data presentation. Thus, client-server computing replaces the centralized computing paradigm.

In the client-server relationship, processing is split between the client and the server. The client systems run an application that displays an interface for the user. It formats requests for network services and displays information or messages it received back from the server. The server performs back-end processing, such as storing data or performing extracts. Because the data is close at hand, it performs this processing efficiently. After storing, or performing some other service on data for a user, the server sends the results back to the client. Network traffic is reduced because the client only gets requested information, not large blocks of data to sort through.

In the light of the new security hardware and software equipment, the implementation of security becomes an easy task. Although, the existing access tools have simplified the hacking process. This part of the thesis is discussing the security gains in the light of the proposed IE design which cannot be considered as a security policy or standard. In fact, it is only dealing with securing the different database elements taking into consideration the treelike breakdown of the cooperating database systems.

Because of the heterogeneity in both the hardware and the software surrounding the database systems the security enforcement requirements has increased. Unless the heterogeneous data sources could be accessed using a unified access methodology regardless of the database system design it would be very much difficult to deal with the individual cases. The IE designs' considerations do not only fit information sources with different schemas and multiple platforms but satisfies the requirements for the similar distributed database applications with different schema designs.

Ensuring security means preventing, detecting, and deterring the improper disclosure of information. In database environments, the different applications and users of an organization refer to either a unique integrated set of data or different types of databases through the DBMS. In the first case, there will be a single security manager since it is a single type schema. The existing scenarios for the second case are different security managers supporting different schema types. In this second situation if already bridges defined between the different schema types then security rights for the users of the different schema types have to be defined by each of the security managers of the different schemas. This process may not be time consuming and may increase the threat on the databases because of the different policies and standards that may be followed by each of the information sources.

The increasing uses of large multiple access data systems and distributed database systems increase the risk of unauthorized use. Information security is therefore an important research area. This includes both access control systems and security services based on cryptographic machines. Evolution of IT security internationally recognized by research realms as a strategic activity has to be undertaken in a way to create standards and procedures for the implementers to follow. Such process will definitely increase the productivity between the information producers and consumers in a systematic and secure manner.

In order to protect the database, security measures must be taken at several levels. Those levels can be ranging from the hardware to protocols and coming down until reaching the databases, which is the main core of this research. Database security management systems are basically those systems used in the definition of the database security levels, ranging from user authentication to assigning different access privilege to different users. The definition of security levels can be defined for either individual users or sites. Also this subsystem will support profiling techniques so that profiles on local data stores are created and individual users profiles or sites can be assigned as part of the profile. By this step, forward planning of system-wide security, especially for the distributed databases that are created as a result of external factors such as a corporate will help in the elimination of the problem.

Work on secure distributed databases continues in areas such as developing an architecture capable of supporting both local and global multilevel processing, which is ranges from the whole database to certain specified columns or even rows. Another research area is security in heterogeneous multidatabase systems. The extent to which powerful multilevel security will ever be successfully applied to corporation-wide multidatabase environment is uncertain, given the many complexities in that area such as the level of the taxonomy of the multidatabase system as shown in Figurer 4.1, Chapter 4. Also, the widely varying requirements for data protection are challenging areas and need to be dealt with by creating a unified high level standard in security management. In addition, many security management independent islands have been created because of the differences in issuing security constraints between the different DBMS vendors. A unification process for such taxonomy will help in many areas as will be explained later in this part [Simon95, Jonscher95].

Interoperating between underling security services may touch many areas such as authentication, access acquisition, key distribution, certificate management, and audit. For example, key distribution services may need to communicate with each other, and audit services may need to transmit audit records between systems.

10.1 Background of the problem

Multilevel security in the relational distributed databases means certain users can access subset of certain table. Hence, the security can be given on certain rows of the table, certain columns of the table, or both. Many security services have been designed and implemented on many different platforms. However, these implementations are often not compatible [CCITT].

Security is still a major problem failing in current DBMSs. Heterogeneity and distribution make this open problem even more difficult. A database owner may want to make certain databases or part of them only accessible by certain users. However, databases should only be accessed by authorized users [Siberschartz94]. Security in general requires the following interrelated components secure:

1. The intercommunication lines and protocols, (physical, data-link layers)
2. The intra-communication lines and protocols, (physical, data-link layers)
3. The network operating system, (application, presentation, session, and transport layers)
4. The operating system, (application, presentation, and session layers)
5. The database management system, (application layer)
6. The database management application, (application layer)
7. The data stores, (application layer), *which is the main part of discussion in this research.*

Network security can be seen as a collection of services which: maintain the confidentiality and integrity of the message as well as the network, provide for the authentication of users, and make sure of non-repudiation by users and the non-denial of services.

Because access to the distributed/shared databases is critical, this access has to be authenticated in a way to prevent systems from unauthorized access. There can be two types of authentication exchanges. The first is called a simple authentication by which a very simple authentication process is applied such as the supply of the user id and password, and the recipient checks these. The second authentication type is the strong authentication where by cryptographic techniques is used to protect the exchange of validating information. Usually this type requires more than one set of validating information exchanges to successfully complete the authentication process [Prabhu96]. Today, there are number of strong authentication protocols in use. Examples are Needham-Schroeder protocol [Needham78], based on the symmetric key encryption; the CCITTX.509 authentication protocol [CCITT], requires a control repository of information to store the credentials of the principals; and Kerberos authentication server which was developed in the Athena project [Steiner88] is probably the most widely used authentication service.

Part of the IE analyzed prototype proposal is considered as an added security on the information resource. First, unified views from the actual information source will be derived for the share purposes. Second, additional security assignments could be added to the unified view in order to increase the actual information resource database management system security capabilities. Third, if the information resource lacks the security capabilities then the IE security will takeover the shortages in this area. In the case of warehousing is considered, the security requirements have some similarities, the autonomy on the information source is considered as minimal. The invited talk [Bhargava2000, Ting95] has considered the different security requirements in the data warehousing where this security prototype is considered as an advanced security in the way dealing with the different information source components responsible to formalize security requirements. Furthermore, the analyzed prototype looks to the different distributed information sources willing to cooperate between each other in an individual manner rather than looking at the information systems as a unified component as in data warehousing. Although the security in both, data warehousing and information cooperation, should have high security, the autonomy is minimal or totally doesn't exist in the data warehousing.

10.2 The proposed unified security system solution

As a solution to the heterogeneity in security management system, the security managers in the distributed databases should liaise with each other so that propagation of security constraints is possible between security managers. In this case if the local site knows other remote sites' security constraints, remote requests may be checked locally before traversing and being rejected in the remote site. This will save the requester time and minimize the load on the wide area links.

According to Figure 10.1 the unified security manager will interoperate between the different security management operations. The unified security manager will liaise with all the different security managers in the site so that all the databases operate under unified security manager. This will make the process of the security manager an easier process. Also, the assignments of local and global users will be an easier process as well. And of course interoperability of security management means that through a unified security manager it will be possible to assign access rights on applications, files, and systems. The autonomy in this situation can be defined as the single management point on different security management systems where the access to all the data stores should be through this point.

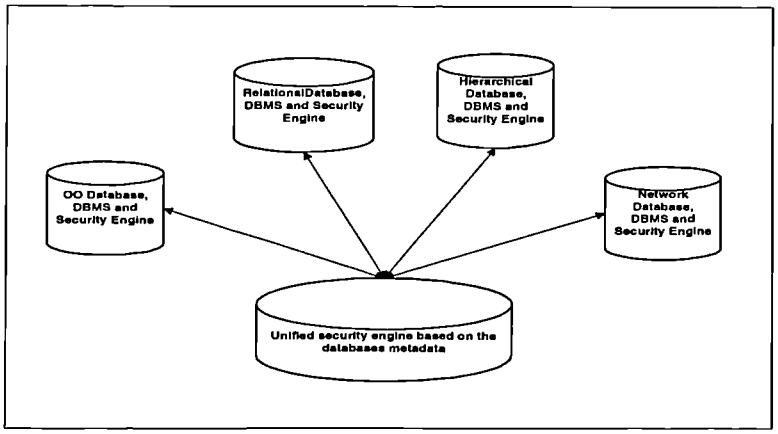


Figure 10.1

In object-oriented databases, the general issue of mapping object security to database security in object-oriented databases is not well defined yet [Mars96]. This issue will be better understood as some implementations of the object security services become available and become integrated into persistence. The object will be stored in a data store, and the security will than be provided by the object security service, which will decide which users may invoke which methods on which objects.

Object provides a clean, realistic model of persistent entities with complex behavior found in most heterogeneous distributed systems. Several architectures, most notably CORBA [OMG91, Mowbray95] and DCE [OSF92, King93] have been proposed a standard for distributed object management systems. However, these lack the formal foundation necessary to verify system security and other critical properties of high assurance systems [Hale96].

As security advancement the IE metadata system could be configured as shown in Figure 10.2. It could be considered as a three level architecture information provider. A naming layer can exist between the local IE information object and the global IE information object to supply the different naming enforcement that would be used by the global users which are originally linked to the original names supplied by the information owner. The naming links objects were considered as part of the syntactic/semantic data dictionary and were discussed in Section 9.1. Now, it is possible to secure the actual database names from being known. Also, the one-way reading constraint forms another security firewall against intruders. The implementation of such constraint is considered as a simple task in the light of the rapid improvement in the security and firewall related equipment.

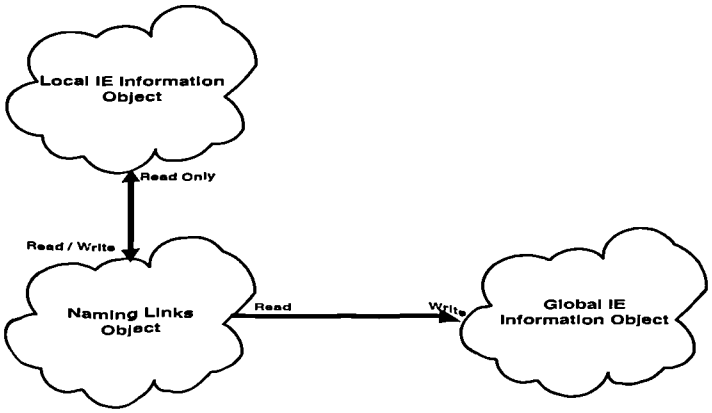


Figure 10.2

Chapter 11

Existing Middleware Solutions Need a Middleware

The goal of the Interoperation Engine IE is to provide fairly transparent views for the heterogeneous information sources to enable them to interoperate easily. The IE project is mainly a development exercise for a set of tools based on policies and standards that facilitate the interoperation process in a cooperation manner for the heterogeneous information sources developed based on the structured database schemas. This chapter presents an overview of the project, describing the various components of the project and the interrelationship between them taking into consideration the following seven areas:

1. Initiate a web capable browsing facilities for the heterogeneous information sources willing to cooperate using the metadata of those information sources.
2. Initiate the design for a universal metadata viewing system capable of interoperating between the heterogeneous information sources in a manageable manner considered as an additional component in the current Internet browsers.
3. A filtering and search engine to browse the metadata information in the Internet browsers.
4. Establishing the necessary linkage to the heterogeneous information sources using the metadata browser facilities.
5. Apply a new security layer to the information sources owner in the web.
6. Provide a unified way of contact between the information producers and consumers.
7. Better usage of the processor time by having the required business rules and the management of the different constraints to be met in the metadata knowledge base when the request initiated by the information consumer.

The cooperative interoperation in the IE could be best described by the mutually benefit relationship between number of information sources in terms of exchanging information and using each other information results in a cooperative manner. To expand the information samples it is most desirable to get similar information from others and set a cooperative share with them. The only obstacles behind this scenario if the others information sources either using different schema type and/or is fully designed different from my local information source.

A well-known common problem facing many organizations and decision-makers is the heterogeneity, disparity and multiplicity in the information sources they are willing to get information from them. Some of those sources are object based and others are relational. While others are legacy based on hierarchical and network schemas.

In open systems area, several standards for the open systems has been issued such as SQL Access Group, ANSI/X3H7, OMG ODMG, Microsoft ODBC, IDAPI, and Borland ODAPI whom are striving to improve the openness of the future database system and its applications. The trend will be toward designing global applications that are running on the client side system and at the same time capable of drawing data from variety of servers with standardized data access protocols.

On the other hand, database management systems are mainly consisting of number of roles built to manage number of components in an organized manner. These components are the GUI front end, the processes that carry the business roles of the database application, the memory management to get and put the record components to and from the permanent storage and lastly the storage and the accompanied management processes. Basically the DBMS bind these components together. The last two components are mainly the DBMS local operating system capabilities. The DBMS works here as a custodian to take over the management required by the memory and the storage.

Most of the application buyers are looking into the application ability side rather than the database management systems' capability. This is due to the selection of the application which is driven by the application availability and suitability. This may be true for some cases and may not lead to discrepancies if their business objectives are not to integrate their information with other information sources in order to effectively use their IT resources. It is obvious that today's standards and available products do not meet what the customers are looking for. Nor it is economical in terms of the requirements when consider the support and the problem management costs. So again it becomes obvious that the open system's promises are not yet fully met. Especially in the sense that the current database requirements are to provide an internet-enabled databases for the sake to interoperate with others information sources.

End users wanting to interoperate by integrating their desktop tools and applications in a consistent and managed manner with whatever databases laying in the enterprise in even a concurrent manner. This is not yet easy because of the different dialects in SQL, gateways, and the other required middleware solution components. A unification process to the views of the information sources heterogeneous schemas that are willing to interoperate has to be considered. Such requirement would add another layer, which to an extent will form a burden, sense it is considered as an additional dependency on the interoperation while its advantages are considered important. It would form a unified atmosphere for the heterogeneous schemas, which would be considered as another layer of security on the information sources.

One of the very well known complexities of the relational model is its viewing feature. Updating of views is complicated and can be ambiguous. In general, update on a view defined on a single table without any aggregate functions can be mapped to an update on the original underling base table. Additionally, for a view involving joins, an update operation may be mapped to update operations on the underlying base relations in multiple ways. In general, the topic of updating views is still an active research area that needs to be looked at from an open elevation.

In an open manner, and from the extensive research that has been conducted in the different schemas available around the world it is found that the object orientations is nothing other than an amalgamated picture of all the existing database schemas. It seems that the ways do analysts and designer thinking has driven the idea of object orientation. This is why it could easily be proved that any designed schema could be converted to any other schema type. For example, it is intended to simulate all the schema types using the relational capabilities and this, of course, prove the visa versa.

As stated in many of Date's papers [Date98a, Date99a, Date99b, Date99c, Date89b] the considerations that have been made so far for the object-oriented schemas are originally the cores of the relational schema. In other wards, there is nothing that cannot be handled by the relational schema if it is according to the initial outcomes of Codd's original papers. Even though, according to the survey study conducted in the heterogeneous database schemas, that all the object-oriented based features are applicable by the existing relational schema. Examples of conversion applicability are reported in [Grimes98, Klas94, Ambler2000, Qian95, Keller97]. In addition, [Calman94] has surveyed different issues in both the object model and the relational model and concludes with assuring the capabilities of the relational model in doing all the proposed object-orientation operations.

11.1 Current database middleware solutions

Existing primitive middleware solutions may describe a handful of products that use radically different architectures to offer broadly similar sets of services. These products mediate high-level communications between client and server or server and server and can be divided into five primary categories:

- Message passing systems.
- Remote procedure calls (RPCs).
- Object request brokers (ORBs).
- Online transaction processing (OLTP) monitors.
- Database management systems (DBMS).

Also, the current existing heterogeneous database middleware solutions are totally configured statically and customized to solve specific requirement. Furthermore, the existing solutions do not have added facilities on the databases such as fault tolerance techniques in case disconnection happens between the different related information sources, unified security management, transaction management, etc. The main dependence on providing such facilities is only on the DBMS capabilities where the original information source has been crested under it by which some or most do not have such facilities. Figure 11.1 shows the current Microsoft solution scenario on this behalf for accessing a mainframe legacy information source.

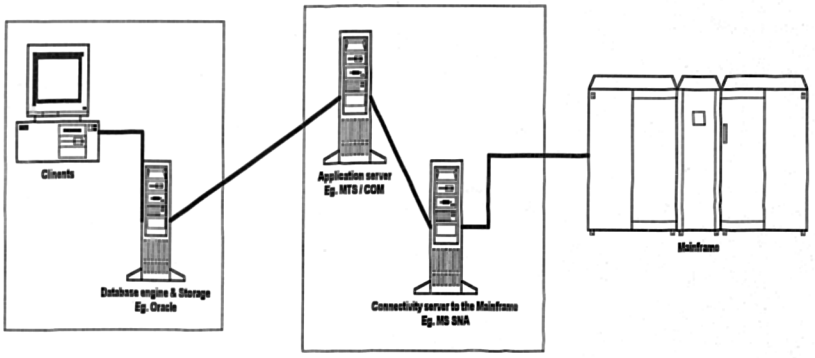


Figure 11.1

The existing solutions are dependent on the requirements provided by the information consumers and the access rights given by the information producers. Hence, the producers need to set up wrappers on their application side to retain their local business roles. If the application is legacy there will be no way the information owner can give access on part of the information source other than the defined in the application unless he writes another code with the specified requirements and wraps it separately. Furthermore, if changes in the business requirements are to be carried out on the application this will require changes to be carried out in the client side that accesses the wrapped code. Also, multiple code parts may be required for the different access cases. By this situation it will end up with a hassle caused by uncontrollable wrapped codes and different middleware engines and different requirements in terms of the required parameters.

Although the wrapping technology has dramatically contributes in solving many of the downsizing/data migration requirements, it appears, recently, that it is still lacking contribution in important complementary related areas to the information interoperability such as fault tolerance enforcement and security synchronization. Also, code-wrapping technique may work properly for the fully structured written applications where the business roles are written properly. While it will require rewritten to some or most of the application code parts and wrap them separately in systems lack proper structure design and proper business roles assignment, which is the case with the majority legacy database applications.

The most important here are the added layers towards creating any linkage between the dissimilar information sources. In that the added layers will either slowdown the process or they may have conflicts because of the special purpose they been built to meet. In addition

the management of the different layers may become difficult sense the changes to meet the different cases need to be done statically to meet the individual cases requirements. The most ideal and accurate solutions are those taking the information from the actual information sources rather than from sources created by mediators where at least the information peas may not up to date.

Open database system means others can use them in an organized manner. The database would not be considered open if it does not advertise over a popular network such as the Internet. At the same time the information resource, which is the database, should be fully controlled by its owner. Also, any later changes in the information sources structure should not effect the other information consumers having a link with this information sources unless drops in the information source is applied by the information source owner.

11.2 The IE prototype

The work in the IE is planned towards creating the suitable environment for the heterogeneous information sources willing to cooperate by exchanging and sharing their information with each other. As been clarified in the introduction, database management systems are simply tools managing storage pointers between both the memory and the storage. The tasks of memory and storage location assignments are totally taken from the operating system where the DBMSs are working under them. And, sense the DBMS works under the local operating system, then it should fully synchronize the memory and the storage addresses assignments with its local operating system.

The main problems of the Universal Object Identifier (UOID) could be considered in the case of the services are only between objects, while in the case of information sources a different universal identifier is required. This identifier should take into consideration the different information sources components so that the record or object is accessed from anywhere depending on that universal identifier. In this case the generated database view, which is assumed to be an IE format standard, has to apply on of three cases to stay in the cooperation. First, the view has either to stay forever while the cooperation takes place. Second, the key part related to the attribute is a driven from the attribute value so that it can be regenerated again on the same value when required. Third, to have an additional space in the original schema of the information source that is updated with the key generated for the cooperation purposes. Since this proposed universal identifier is directly adhered with the record object and not related to the storage, rebuilding to the information sources will not be affected. Also, for the purpose of accessing the object record in the storage there will be another complementary object created temporarily when rebuild to database is applied having the object identifiers and the physical storage addresses.

Now, what are the likely effects in case the URL location of the table is added to the indexes files applied on the database information sources sense again rebuild will change the physical location of the record? Again, such situation will require another hashing like table plays as a mediator resource between the logical record object and the physical location of that object. Such situations will weakness the possibility of having the physical record object address in the index. Even though careful implementation of such technique will increase the performance of the information retrieving.

Here we shouldn't mix the memory operations and the storage operations. In case if the record in the index forms a pointer to a remote record then it should carry the ID of a record in a table in the remote database/table side having the actual physical memory/storage of that record. The data structure used to model the various index organizations is the binary tree B-tree [Dynamic99, Korsh88, Wiederhold88].

The database information systems, whether they are from the legacy or new systems, are mainly built out of three main components. The first is the repository, which is a standalone place, where the information is gathered according to certain criteria. The second is the business rules that are defined by both the data definition phase and within the application that derives the repository. The third is the process sequence, which is considered as part of the business rules that is driven by the application and probably by the schema in the data

definition. If the repository is object oriented and not a standalone as in the relational case, hierarchical and network models other than the first and second components are usually embedded together.

Most of the existing legacy database information sources residing around the world are mainly designed using the primitive analysis and design methodologies. At this time none of the analysis and design methodologies are considering information interoperability between the heterogeneous information sources. This part has been recently defined by the object-oriented analysis and design methodologies such as the one come with the OMG ODMG [Busse94].

The middleware solutions built for the sake of information interoperability will carry some discrepancies since they are accessing the database sources through the application that accesses the data source itself by assigning code wrappers. This is because the source application simply defines all the business rules, as well as, the different attribute constraints. Of course, different hardware will require different middleware setup with special preparations related to the operating systems. In the light of the current requirements in the information interoperability area, the middleware solutions could play a big rule in area of database connectivity. The rest of the interoperation requirements need to be assessed by a middle interoperation layer capable of simplifying the mission for both the information producer, as well as, the information consumer. This area is the subject of this chapter that is an explanation of its breadth and the likely proposed solution towards a unified interoperability layer accompanied by a unified interface for such purpose.

Two different groups of users of the IE are assumed. In the first group, users accessing information sources from different information producers' sites using the IE. The second group has the same purpose as the first but unifying the information sources with their own local information source using the unification facilities provided by the IE.

The mechanism for a view, which is the IE prototype basement, is an important discretionary authorization mechanism in its own rights. For example, if the owner A of a relation R wants another account B to be able to retrieve only some fields of R, then A can create a view V of R that includes only those attributes and then grant SELECT on V to B. The same applies to limiting B to retrieving only certain tuples of R; a view V' can be created by defining the view by means of a query that selects only those tuples from R that A wants to allow B to access.

11.3 The IE different processes

In this section, Figure 11.3 shows the route map of the different interoperation processes to be undertaken by the IE. It is assumed that all the different database components object identifications will depend on the unique machine access card MAC address. This assumption is only true if the MAC card does not change as long as the provided shareable information sources by this machine rune. In the case when the access card becomes faulty then the IE will be responsible to inform the other distributed information consumers about the replacement. The plan in this case is the IE will provide an exceptional table to redirect the sharing information packets to the correct addresses. The smallest database component is considered as the attribute. The proposed IE universal attribute identification was shown in Figure 9.5.

Also, the plan is to have all the related database components that are higher than the attribute to be generated in the time of information delivery. Referring to Figure 11.3 the following steps will be undertaken for the interoperation and cooperation processes in the IE:

1. Metadata registration in the local IE knowledge base side.
    - Read the metadata definition from within the IE. This facility should take the different schemas data definition code and be able to scan the code and update the IE databases with the different definition values. Afterwards the metadata code be altered or dropped.
    - Manually substitute the values to the IE. Such option is usually selected in case minor part of the information sources would only be substituted to

the global. This mode only need to be used when we initially willing to create the data definition script of the information source using the IE capabilities. Or it could be used when only minor part of the information resource is to be defined for the global or local users

2. Selection of the shared parts of the complete information sources from the local IE side.
   - A name to the sharable sub system part is given.
   - Define all the related tables to the defined sub system name. The table at this stage is equivalent to the relation schema in relational model terms; record type in the network and hierarchical models and class description in the object oriented model terms.
   - The IE system at this stage will automatically pull up all the related attributes to the tables by which an exclusion to some of them, according to the information sources owners' requirements, can be done in this step. The IE will make sure at this stage that the exclude attribute are not either primary keys or foreign keys they could have links with other tables. One of the objectives of the IE layer is to build tables to meet the global information cooperation requirements. It is meant by this that during the process of the table creation if it is done through the IE layer a consideration to a tailored tables could be taken while first the metadata of the table is created and second while the individual records are written to that repository.
   - Additionally, at this stage we could define the permitted local users and user groups on this shareable subsystem if the access for the local users is to be controlled by the global IE. In the case of the shareable subsystem being only for the local users and no cooperation been done with an equivalent global information sources this step should not be undertaken.
   - Also, at this stage a name for the shareable subsystem has to be defined prior to transferring the metadata to the global IE knowledge domain so that the subsystem is known and accessed globally by this name.

3. Transferring the selected information source part from the local IE knowledge domain to the global IE knowledge domain. At this stage still the information source owner can define more access restrictions based on the global users and users profiles over his own data. For each access space an XML script and the related necessary buffering environment will be created. The XML script will be based on the relationship between the different classes as discussed in Appendix A.

4. Normally the transferred metadata information will contain the access rights on the different information source components at the time it gets transferred. Additionally, from within the global IE interface an additional security parameters can be assigned before activating the metadata to the global information consumers.

5. Activating the shared information sources to be used by the global information consumers. Only at this stage advertisement about the available information space will be know for the global users. Also, at this stage the propagation of the advertisement can also be for a selective users and/or user profiles. This stage proofs the full autonomy ownership on the information source by the information owner. Since the information owner will be able to selectively update the individuals IE knowledge bases by the information space and its description he can provide them with. In addition, at this stage we can specify the permitted users and user groups from the global IE knowledge about the global users having similar area. This step is considered as the cooperation process between my information source and other similar information sources define the links to enlarge my information area.

6. A reassignment to the access rights after the last stage can be done and a reactivation to the modified parts can be propagated again to the information consumers. Because changes at this stage have taken place at the local site, other involved information consumers have to accept the changes or otherwise they won't get the information they ask for. This stage approves the full autonomy on the information sources to owners of the information source.

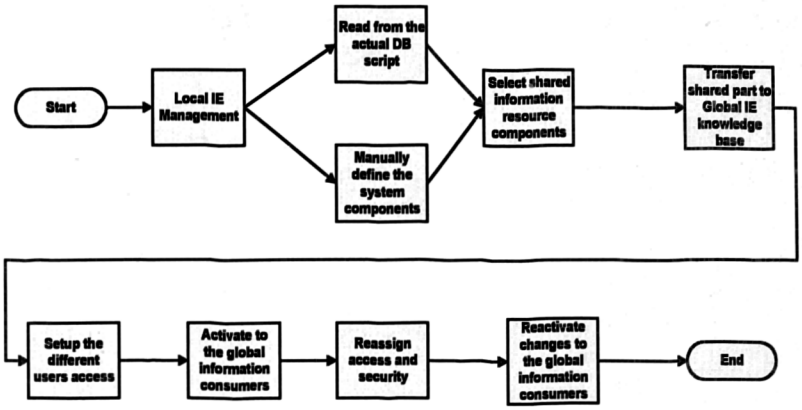


Figure 11.3

Despite there are many middleware providers in the IT market these days, the majority has not provided the answers to several problems. The status becomes harder when legacy applications with a legacy code pages comes in light. Conversion of code pages becomes a major part in today's middleware solutions. In particular when converting characters in languages such as Arabic, which requires in some cases to convert one character in one side with two characters in the other side.

The most important issue in middleware solutions is the robustness of the provided solution. This is in terms of the simplicity of usage, maintainability of the shared information sources parts, availability of information for the global and, of course, security of the provided information sources. Here all the considered facilities in the proposed solution have been fitted. This work is considered to be a complementary to the existing interoperability solutions.

Currently, the IE is considered as a standalone system which is mainly working at the same level of the DBMS (i.e. under the operating system) where the checking is done using the IEs' checking procedures. Eventually a further study to the possibility of embedding the IE on the operating systems kernels is considered. Such a new operating system enables users to manage the heterogeneous information sources interoperability process will be initiated and searched.

Chapter 12

The Interoperable Engine User Interface

Vincent Quint the *User Interface Domain Leader* said "Web information will grow immensely in variety, and be used by a much greater diversity of people than today. What is imperative is that simplicity and interoperability continue to be of prime importance". User interface research groups seek to improve all user/computer communications on the Web. In particular, they are working on formats and languages that will present information to users with more accuracy and a higher level of control.

User interface standards have become the object of increasingly intense activities in recent years [Abernethy88, Holdaway89], including work at the International Standards Organization (ISO) and the European Community [Stewart90]. Research is also going on at national standards organizations and in several major computer companies. Results showed that there is still no unique standard which can that is followed in designing user interfaces. Instead, partial standards are mixed with requirements and tastes. Close study to the various user interface standards will be conducted soon as part of the requirements for developing an easy user-friendly interface for the IE.

The original Interoperable Engine (IE) design components have been prepared to serve a closed business site with local schemas interoperation as well as the interoperation with other global schemas. In order to provide maximum security to the information consumers' data sources, it is strongly suggested by the databases and security designers to segregate the paths of accessing the databases for both the local and global users. It is known from experiments that the performance of local systems is much better than the performance of global shareable systems.

Each IE environment site consists of three administrator user types. These are the administrator who has access to both the local and the global IE sites, only to the local IE environment and to the global IE environment. Each local IE and global IE is considered as a standalone environment. Each has the full options that are supporting the database interoperation requirements. The only connectivity between the local and global IEs is the transformation of the cooperating information sources metadata from the local IE to the global IE and not vice versa.

This chapter mainly describes the IE design of the interface accessible by the global information producers and consumers which forms part of the World Wide Web. In particular, the following functionality was taken into consideration during the design steps of the IE to take place in the different required system interfaces.

Information producer metadata registration:

- Preparing the whole information about an information source.
- Transferring the shared parts of the information source.
- Setting the access rights on the information source.
- Editing a previously registered information source.
- Propagating the updates to the global users.

Information consumer facilities:

- Browsing the available information source.
- Binding the local information source with the other available global information sources as a single unified database.

- Browsing the access rights of the available information sources for only personal account and any account where permission was granted.

Database Administrators facilities:

- Reassignment of the access rights and information sources sizes to their local information consumers.
- Binding the local information source with the other available global information sources as a single unified database for a group of consumers by passing them to pre created profiles to access the various information sources transparently.
- Make the necessary bindings between the local data sources and global data sources.
- Distribution and redistribution of the access profiles to the local users.
- Assignment of access rights to the local and global users in his domain.

The main focus point during the design of the IE prototype is the information cooperation process that involves getting information from many disparate databases with different schema types. Considerations such as remote updating form and to different schemas are taken into account in the IE design. The normal process that will happen is that an information owner who wants to provide access on his/her information to the other users having similar information will make this information available to his/her local IE which will take the responsibility of informing the others permitted users.

The information cooperation process will require number of parameters to be activated by the IE. The following is a list of the prerequisite parameters required for the interoperation:

1. Systems stamp is where the wide area links definition exists. Mainly it is where the Internet Protocol IP addresses and the domain name systems names are defined. Sense the IP is the most common protocol in use these days; it has been used for the communication mechanism between the distributed IEs.
2. Systems path is the set of parameters where the actual systems' information is stored. With the systems' stamp information it forms the exact path for the incoming and outgoing processes.
3. Shareable subsystem is the name of the system which shares information with other global systems. Each system's stamp and path may have more than one shareable subsystem at a time.
4. Schemas are the actual tables belong to certain shareable subsystem.
5. Attributes are the number of fields belong to a schema.
6. Constraints are the list of restrictions that should be met prior to any updated to the actual information sources. Normally constraints are assigned to attributes within a table. The lists of possible constraints for different schemas are shown in Figure 8.9.

12.1 System startup

The initial plan for the startup of the IE system with an Internet browser where the IE is considered as part of the browser supporting the interoperability of the heterogeneous distributed databases. A startup interface the database administrator can use it to do the necessary cooperation bindings to users. The administrator also gives access to other global cooperating users on the local site information sources through the IE. The initial main menu page is shown in Figure 11.1. Although the initial screen for both accessing the local and global IE engines has to be an access authorization screen.

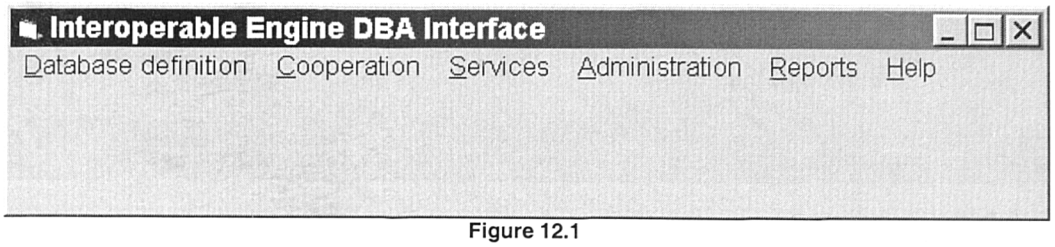


Figure 12.1

The DBA has the full access to the information sources provided for his site. Also, there are some users having partial administration capabilities on partial information sources and they will also have similar interface as shown in the above figure but without the DBA capabilities. In general, the initial system interface has following options:

1. Database definition
   - New system
   - Edit existing system
   - Delete system
2. Cooperation
   - Add new project
   - Edit existing project
   - Delete project
3. Services
   - Import information sources to the local IE
   - Export local information sources to global IEs
   - Cooperation propagation
   - Cooperation setup
4. Administration
   - Set access of individual users
   - Set group access on information sources

12.2 Information producer metadata registration

The initial Figure 12.2 demonstrates the layout of the information producer interface, which will be used in registering the metadata of the different information sources components.

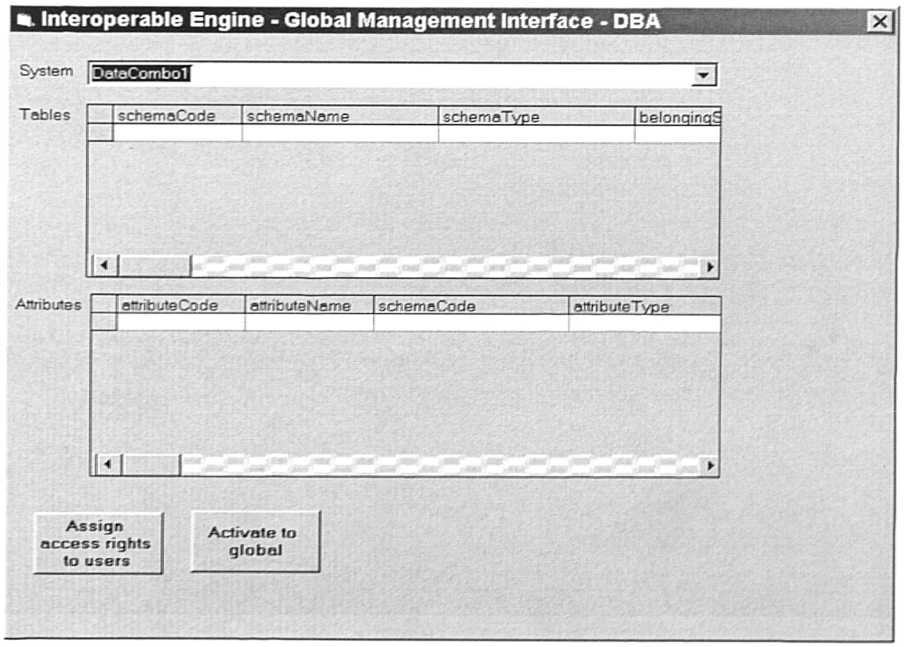


Figure 12.2

12.3 Information consumer facilities

The information consumer can either use the producers information sources as it was provided by the local IE layer or to merge the information sources with his/her own using two different approaches. Either accumulate the provided information sources with his/her own information source in case it has similar information or a relationship with that information source could be established. On the other hand, the full IE system has the following interfaces:

- Client Interface
- Server DBA Interface
- Local DBA Interface
- Cooperation Data Sources Query Services

The remaining interfaces such as what will appear on the browser were shown in Figure 8.2, Chapter 8. Also, throughout the explanation of the different sections in this thesis a formal explanation has been done to the different prototype components.

12.4 Database Administrators facilities

Normally administrators are the highest people in terms of the access rights on and of the systems in the world. They have the authority in giving other users the rights on certain systems as they have or even segregates people from accessing certain systems. The same case is applied to the IE by which they may give partial cooperation implementation capabilities to certain users where they can build their own cooperation services in between their own information sources and others information sources. Or they may build the cooperation profiles by which they can grant usage access to different users and users groups. Dealing with the different IE user interfaces the following steps shown in Figure 12.3 have to be followed for the different cooperation processes.

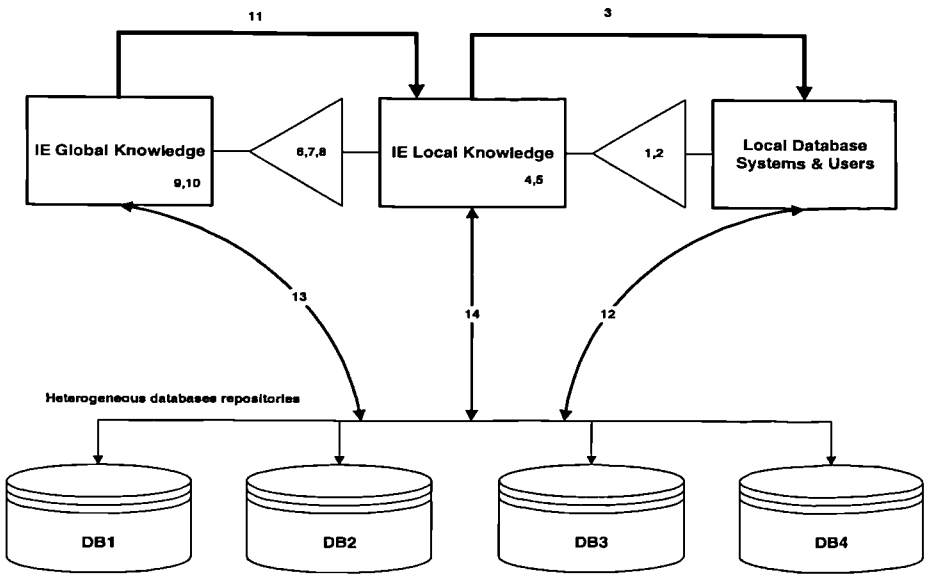


Figure 12.3

1. The local users information is pulled on the local IE knowledge base.
2. The database systems metadata information could be transferred from the local database systems area to the local IE knowledge base.
3. Define the metadata of the database systems and push to or read by the database management system. The specific definition of the schema type and the database management system name is required in this case sense each DBMS have its own metadata definition criteria.

4. In the local IE system the metadata of any database system can be defined and the script for creating the actual database can be also generated.
5. The definition of the different shareable sub systems are defined in the local IE repository prior to pasting them to the global users area in the global IE system knowledge base.
6. Originally the local systems users are residing in the local network operating system or the local database management system. After achieving step number 1, the users may again be regrouped and passed to the global IE knowledge base prior to attaching them to shareable sub systems.
7. The shareable sub systems metadata with the different assigned permissions is done at this stage. This is considered as the green light to the global users to access the local information sources. From this stage the site DBA can propagate the necessary information to the distributed IE telling them about the available information space.
8. Any updates to either the users or the shareable sub systems will be done at this stage. The updates are adding new users, deleting users/groups, change in the access rights on certain shareable sub systems will be done through the local IE knowledge base and will be transferred to the global IE area by which it will propagate only the updates to the global IEs. The proposed handling protocol discussed at section 8.4 will takeover this process.
9. The IE global knowledge mechanism deals with the local information sources when queries are submitted to it.
10. In addition, the IE global knowledge base will assign and reassign the different access rights on information sources additional to the ones come from the local knowledge base.

The remaining of steps are those related to the permitted access between the IE layers and the physical information sources. As shown in the figure, the various verifications for the IE information will directly be confirmed from the actual information sources.

Chapter 12

The Interoperable Engine User Interface

Vincent Quint the *User Interface Domain Leader* said "Web information will grow immensely in variety, and be used by a much greater diversity of people than today. What is imperative is that simplicity and interoperability continue to be of prime importance". User interface research groups seek to improve all user/computer communications on the Web. In particular, they are working on formats and languages that will present information to users with more accuracy and a higher level of control.

User interface standards have become the object of increasingly intense activities in recent years [Abernethy88, Holdaway89], including work at the International Standards Organization (ISO) and the European Community [Stewart90]. Research is also going on at national standards organizations and in several major computer companies. Results showed that there is still no unique standard which can that is followed in designing user interfaces. Instead, partial standards are mixed with requirements and tastes. Close study to the various user interface standards will be conducted soon as part of the requirements for developing an easy user-friendly interface for the IE.

The original Interoperable Engine (IE) design components have been prepared to serve a closed business site with local schemas interoperation as well as the interoperation with other global schemas. In order to provide maximum security to the information consumers' data sources, it is strongly suggested by the databases and security designers to segregate the paths of accessing the databases for both the local and global users. It is known from experiments that the performance of local systems is much better than the performance of global shareable systems.

Each IE environment site consists of three administrator user types. These are the administrator who has access to both the local and the global IE sites, only to the local IE environment and to the global IE environment. Each local IE and global IE is considered as a standalone environment. Each has the full options that are supporting the database interoperation requirements. The only connectivity between the local and global IEs is the transformation of the cooperating information sources metadata from the local IE to the global IE and not vice versa.

This chapter mainly describes the IE design of the interface accessible by the global information producers and consumers which forms part of the World Wide Web. In particular, the following functionality was taken into consideration during the design steps of the IE to take place in the different required system interfaces.

Information producer metadata registration:

- Preparing the whole information about an information source.
- Transferring the shared parts of the information source.
- Setting the access rights on the information source.
- Editing a previously registered information source.
- Propagating the updates to the global users.

Information consumer facilities:

- Browsing the available information source.
- Binding the local information source with the other available global information sources as a single unified database.

process should hide the differences in all the stages of interoperating any two information sources. In general, information interoperation is a broad meaning for information cooperation where cooperation is the process of sharing information sources or parts without loosing the autonomy on the resources.

Database interoperability problem is complex, diverse and lacks a single standard. Support from a higher level than the database management system is required to find a general solution to this problem. Focus on the problem of the heterogeneous database interoperability need to be started.

Also, the current Internet search engines are not considered as database metadata capable. They are handling normal HTML metadata and links will be established with pages developed using the HTML language. On the other hand, information consumers are more interested in metadata capable search engines supported by their machine browsers where the information source they can get access with is known. Also, browsers that can easily link them to the information source they want to merge with their own information source are needed.

Information consumers should be able to place a search key to the search engine, get the result, dig deeper in the returned result to get information about the information source different components (supported schemas, related attributes to each of the schemas, relationship and access rights), select and link the others information sources with his own information source.

This research has mainly studied the problem, design an interface for the four known schemas (relation, hierarchical, network and object-oriented) and search the various areas that will support database interoperability (security, transaction management, data dictionary, tracking management, etc.) and find out the linkage between the proposed interface and those areas.

Throughout the research in this area it is found that a number of indispensable services would be counted on the success grade of the information interoperation:

1. A security management services responsible to secure the heterogeneous information sources at any one site is a must. This will minimize the security tasks on security manager.
2. In a heterogeneous information sources environment, clarifications of the different used terms by different people would be very important in resolving the conflicts arises from such area.
3. History tracking to the users' queries would help to restart the uncompleted queries from where they have suspended.
4. A sophisticated transaction management dealing with hits on different information sources needed.
5. Unified indexing is a proposal towards motivating using a universal key in the indexing technique to cooperate in solving the slowness problems of retrieving the records.
6. Replication of information sources is a complementary service; the local DBA will be able to replicate any information source into a second location, which will form backup security and load balancing.

13.2 Results

The results of this research have both direct and practical impact on the information sharing between the heterogeneous information sources. The following are some areas where we have realized the impact:

- A framework for interoperating between heterogeneous and distributed information sources in a cooperative manner has been established. The selected way is considered as dynamic, incremental and autonomous at the same time.
- Architecture based on the number of standards and based on that applied number of experiments to prove the various prototype ideas has been introduced.
- Throughout this research an extended attention has been paid to the autonomy of the individual information sources and approve the ability of having the full rights on the information sources by their owners.

13.3 Ongoing and future work:

The work on the Interoperation Engine IE continues. The proposal is still in its early stages and there are plenty of tasks to be undertaken next in order to have a solid foundation where we can start. The initial design of the prototype is built around number of presumed hypothesis. The first assumption is that the prototype is based on four database management system schemas, which are the IMS, IDMS, Relation and O2. The second assumption is that the operating system and the communication protocol heterogeneity problem are not of any concern in the light of the Internet communications advances. The third assumption is that the IE prototype is the mediation process between the users and the local heterogeneous databases. The following are the ongoing activities towards building the new proposed prototype.

1. Fully implementing the current analyzed IE system.
2. Implementing a syntactic/semantic supporting data dictionary that will takeover the different required clarifications and syntax conversions.
3. Analyze/design the incoming/outgoing messaging and handling subsystems.
4. Study the different XML recommendations [Bourret99, W3C98] to see the capability of making use of them in the different required handling processes.
5. Implementing the shareable data sources profiling subsystem.
6. Using the shareable data sources profiles by the database application.
7. Study the required design changes in the database applications willing to interoperate with the distributed heterogeneous databases.

Although a number of approaches has been studied in this area we are also in the process of studying number of existing approaches to make sure that the proposed prototype, which is assumed to be an extension to them, does not have any major conflicts. In addition, we are in the final stages of experimenting part of the ideas presented in the thesis on a legacy mainframe information system based on the hierarchical database and another relational database system.

Finally, as part of the future implementation commitments in the area an investigation and carrying out a of tasks such as

1. Schema Translation Management for the heterogeneous schemas.
2. Dynamic access management to take care of the incoming requests to its local IE, which will take care of the local process evaluation.
3. Transaction Processing Management, which deals with multi-user transaction management and time slicing.
4. Programming language translation management when only queries are passed from different query systems.
5. And query Decomposition Management.

Reference List

[Abernethy88] C. Abernethy, (1988). "Human–computer interface standards: Origins, organizations and comment", In Oborne, D.J. (Ed.), *International Review of Ergonomics*, 1988.

[Alonso91] R. Alonso, D. Barbara, S. Cohn, "Data Sharing in Large Heterogeneous Environment", Proceedings Seventh International Conference on Data Engineering, April 8-12, Kobe, Japan, 1991.

[Ambler2000] S. Ambler, "Mapping Objects To Relational Databases", July 3, 2000, http://www/AmbySoft.com/mappingObjects.pdf.

[Anderson93] M. Anderson, Y. Dupont, S. Spaccapietra, K. Yetongnon, M. Tresch, H. Ye, "FEMUS: A Federated Multiligual Database System", "Advanced Database Systems", "Springer Verlage", 1993.

[Ashir2000] J. Ashir, "Technical Perspective on the Heterogeneous Databases Interoperability", IEEE Computer Society Press, Proceedings of the 11th International Workshop on Database and Expert Systems Applications, DEXA 2000, Greenwich, London, UK, 6-8 September 2000.

[Bacon93] J. Bacon, "Concurrent Systems", Addison-Wesley, 1993.

[Baker90] K. Baker, "Transaction Management on Multidatabase Systems", Doctor of Philosophy thesis, Department of Computing Sciences, University of Alberta, Edmonton, Alberta, Fall 1990.

[Barbra91] D. Barbra, C. Clifton, "Information Brokers: Sharing knowledge in a Heterogeneous Distributed System", Proceedings Seventh International Conference on Data Engineering, April 8-12, Kobe, Japan, 1991.

[Bertino93] E. Bertino, L. Martino, "Object-Oriented Database Systems, Concepts and Architecture", Addison-Wesley, 1993.

[Bhargava2000] B. Bhargava, "Security in Data Warehousing", Proceedings of Second International Conference in Data Warehousing and Knowledge Discovery, London, UK, September 2000, 287-289.

[Black87] U. Black, "Computer networks, protocols, standards, and interfaces", Printice-Hall International, 1987.

[Booch99] G. Booch, J. Rumbaugh, I. Jacobson, "The Unified Modeling Language User Guide", Addison Wesley, 1999.

[Bouguettaya91] A. Bouguettaya, R. King, K. Zhao, "FINDIT: A Server Based Approach to Finding Information in Large Scale Heterogeneous Databases", First International Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan, IEEE Computer Press, April 7-9, 1991.

[Bouguettaya94] A. Bouguettaya, "Large Multidatabases: Beyond Federation and Global Schema Integration", Proceedings of the Fifth Australian Database Conference, Christchurch, New Zealand, Global Publications Service, Jan. 1994.

[Bouguettaya95] A. Bouguettaya, S. Milliner, "Co-database approach to database interoperability", IEICE Transaction of information systems, Vol. E78-D, No. 11, November 1995.

[Bouguettaya95a] A. Bouguettaya and S. Milliner, ``Data Discovery in Large Scale Heterogeneous and Autonomous Databases'', Special Issue on Advances in Digital Libraries, Eds: N. R. Adam and B. Bhargava, Springer Verlag Lecture Notes in Computer Science, 1995.

[Bouguettaya95b] A. Bouguettaya and S. Milliner, ''Co--database Approach to Database Interoperability", IEICE Transactions on Information and Systems. Vol. E78-D, No. 11, November 1995.

[Bourret99] R. Bourret, "XML and Databases", URL: http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases, December 1999.

[Breitbart86] [Yuri et al., 1986] Y. Breitbart, P. Olson, G. Thompson, "Database Integration in a Distributed Heterogeneous Database System", IEEE Conference in Data Engineering, pp 301-310, February 1986.

[Bright94] W. Bright, A. Hurson, S. Packzad, "Automated Resolution of Semantic Heterogeneity in Multi-databases", "ACM Trans. on Database Systems", June 1994.

[Brodie95] M. Brodie, M. Stonebraker, "Migrating Legacy Systems: Gateways, Interfaces & the incremental approach", Murgan Kufmann Publisher, 1995.

[Browne93] S. Browne, "Communication and Synchronization Issues in Distributed Multimedia Database Systems", ", "Advanced Database Systems", "Springer Verlage", 1993.

[Budd91] T. Budd, "An Introduction to Object-Oriented Programming", Addison-Wesley, 1991.

[Burleson94] K. Burleson, "Managing Distributed Databases", John Wiley & Sons Inc., 1994.

[Busse94] R. Busse, P. Fankhauser, G. Wolfgangklas, "IRO-DB: An object-oriented approach towards federated and interoperable DBMS", Proceedings of the International Workshop on Advances in Databases and Information Systems ADBIS'94, Moscow, Russia, May 1994.

[Busse94a] R. Busse, P. Fankhauser, E. Newhold, "Federated Schemata in ODMG", Proceeding of the Second International East-West Database Workshop, September 1994, Klagefurt, Austria.

[Calman94] D. Calman, "Moving forward with relational: looking for objects in the relational model, Chris Data finds they where there all the time", DBMS Interview, October 1994, http://www/dbmsmag.com/int9410.html.

[Cardenas80] A. Cardenas, M. Pirahesh, "Database communication in heterogeneous database management system network", Information systems, Vol. 5, No. 1, 1980, pp. 55-97.

[Carey94] M. Carey, L. Haas, P. Schwarz et al., "Towards heterogeneous multimedia information systems: the garlic approach", In Technical Report, IBM Almaden Research Center, 1994.

[Castano95] S. Castano, M. Fugini, G. Martella, P. Samarati, ``Database Security``, Addison Wesley, 1995.

[CCITT] The Directory, Overview of concepts, Model and services, CCITT Recommendation X500, Blue Book, Vol VII, Fascicle V 11.8, Geneva, Switzerland.

[Chen91] J. Chen, O. Bukhres, J. Askary, "A Customized Multidatabase Transaction Management Stratrgy", Database and Expert Systems Applications, 4th International Cenference, DEXA '93 Prague, Czech Republic, pp. 80-91, September 1991.

[Chen94] J. Chen, Y. Hung, "An Integrated Object-Oriented Analysis and Design Method Emphasizing Entity/Class Relationship and Operation Finding", Journal of systems software, Issue 24, 1994, PP 31-47.

[Claude93] F. Claude, K. Roger, "Amalgame: A Tool for Creating Interoperating, Persistent, Heterogeneous Components", "Advanced Database Systems", "Springer Verlage", 1993.

[Coad91] P. Coad, E. Yourdon, "Object-Oriented Analysis", Yourdon Press, 1991.

[Coad91] P. Coad, E. Yourdon, "Object-Oriented Design", Yourdon Press, 1991

[Codd70] E. Codd, 1970, "A Relational Model for Large Shared Databanks", Communications of the ACM, 13 (6), 377-390, (June 1970).

[Codd71a] E. Codd, 1971a, "Normalized Data Base Structure: A Brief Tutorial", IBM Research Report RJ 935

[Codd71b] E. Codd, 1971b, "Further Normalization of the Data Base Relational Model", IBM Research Report RJ 909

[Codd71c] E. Codd, 1971c, "Data Base Sublanguage Founded on the Relational Calculus", IBM Research Report RJ 893

[Codd72] E. Codd, 1972, "Relational Completeness of Data Base Sublanguages", IBM Research Report RJ 987

[Coleman94] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, P. Jeremaes, "Object-Oriented Development, The Fusion Method", Prentice Hall, 1994.

[Computer79] Computer Corporation of America, "A prototype chemical substances information network", Tech. Report CCA-79-19, Cambridge, Mass., August 21, 1979.

[Cox87] B. Cox, "Object-Oriented Programming", Addison-Wesly, April 1987.

[Date89b] C. Date, "The Birth of the Relational Model - Thirty Years of Relational", Intelligent Enterprise, Volume 1, Number 1, October 1998.

[Date95] C. Date, "An Introduction to Database Systems", Addisson Wesly Publishing Company, 6th Edition, 1995.

[Date98a] C. Date, "Thirty Years of Relational: Relational Really is Different", Intelligent Enterprise, Volume 1, Number 1, October 1998.

[Date99a] C. Date, "Thirty Years of Relational: The First Three Normal Forms", Intelligent Enterprise, Volume 2, Number 5, March 30, 1999.

[Date99b] C. Date, "Thirty Years of Relational: The First Three Normal Forms, Part 2", Intelligent Enterprise, Volume 2, Number 6, April 20, 1999.

[Date99c] C. Date, "Thirty Years of Relational: Relational Forever", Intelligent Enterprise, Volume 2, Number 8, June 1, 1999.

[Dayal84] U. Dayal, H. Hwang, "View Definition and Generalization for database integration in a multi-database system", "IEEE Trans. of Software Engineering", Vol. SE-10, Nov. 1984.

[Decker92] D. Decker, V. Hereweghen, F. Piessense, "Heterogeneous intradomain authentication", USENIX Association, September 1992.

[Deen85] S. Deen, R. Amin, et al., "The Architecture of a generalized Distributed Database System – PRECI*", The Computer Journal, pp 282-290, Vol. 18, No. 3, 1985.

[Dogac99] A. Dogac, M. Ozsu, O. Ulusoy, ``Current Trends in Data Management Technology, 1999.

[Dynamic99] Dynamic Information Systems Corporation, "The Art of Indexing", A White Paper, October 1999.

[Edelstein90] H. Edelstein, "Distributed Database", Data Base Management Systems, September 1990.

[Elmasri94] R. Elmasri, S. Navathe, "Fundamentals of database systems", "Addison Wes ey", 1994.

[Fowler97] M. Fowler, "Analysis Patterns, Reusable Object Models", Addison Wesley, 1997

[Frank95] T. Frank, "Object-Oriented Technology: PC-Based OOAD Tools", IEEE COMPUTER, March 1995.

[Garcia94] H. Garcia-Molina, et al., "The TSIMMIS approach to mediation: data model and languages (extended abstract), In the Technical Report, Stanford University, 1994.

[Gligor84] V. Gligor, G. Luckenbaugh, "Interconnecting Heterogeneous Database Management Systems", IEEE Comuter, January 1984, pp. 33-43.

[Gligor86] V. Gligor, R. Popescu-Zeletin, "Transaction Management in Distributed Heterogeneous Database Management Systems", Inf. Syst. 1986, 11(4):287-297.

[Grimes98] S. Grimes, "Modeling Object/Relational Databases", DBMS, April 1998

[Grimsom95] J. Grimsom, "Distributed Information Systems", Theory and Practice of Informatics, 22nd Seminar on Current Trends in Theory and Practice of Informatics, Milovy, Czech Republic, November/December 1995.

[Hale96] J. Hale, J. Threet, S. Shenoi, "A dual framework for high assurance distributed object security", New Security Paradigms 96, Lake Arrowhead, CA, September 16-19, 1996.

[Heimbigner85] D. Heimbigner, D. McLeod, "A federated architecture for information systems", ACM Transaction on Office Information Systems, Vol. 3, No. 3, pp. 253-278, July 1985.

[Hess96] M. Hess, J. Lorrain, G. McGee, "Multiprotocol networking - a blueprint", "IBM Systems Journal, Network Technologies and Systems", Volume 34, Number 3, pp.330 – 346, 1996.

[Holdaway89] K. Holdaway, N. Bevan, "User system interaction standards, *Computer Communications*, (April 1989).

[Islam97] M. Islam, H. Selamet, M. Sap, "A dynamic access control with binary key-pair", Malaysian Journal of Computing Sciences, Vol. 10, Issue 1, pp.36-41, June 1997.

[Jeffery95] K. Jeffery, "Database: introduction to problems", Theory and Practice of Informatics, 22nd Seminar on Current Trends in Theory and Practice of Informatics, Milovy, Czech Republic, November/December 1995.

[Jonscher95] D. Jonscher, K. Dittrich, "Argus: a configurable access control system for interoperable environment", Database security IX status and prospects, Proceedings of the 9th annual IFIP TC11 working conference on database security, August 1995.

[Jung97] I. Jung, J. Lee, S. Moon, "Performance of concurrency control methods in multidatabase systems", Journal of KISS[B], Software and Applications, Vol 24, Issue 6, pp.618-28, June 1997.

[Kalinichenko90] L. Kalinichenko, "Methods and tools for equivalent data model mapping construction", "Advances in Database Technology-EDBT '1990".

[Keimbleton81] S. Keimbleton, P. Wang, "Application and protocols in distributed systems: Architecture and implementation", Lecture notes in computer science, Paul Lampson and Siegert eds., Vol. 105, Springer Verlag, New Yourk, 1981, pp. 308-370.

[Keller97] W. Keller, "Mapping Objects to Tables - A Pattern Language", Proceedings of the 1997 European Pattern Languages of Programming Conference, Irsee, Germany, Siemens Technical Report 120/SW1/FB, 1997

[Kerr92] J. Kerr, "An Introduction to Object-Oriented Programming and Information Engineering", Data Resources Management, fall 1992.

[King] R. King, M. Novak, "Supporting Information Infrastructure for Distributed, Heterogeneous Knowledge Discovery", manuscript.

[King93] W. Rosenberry, D. Kenney, G. Fisher, "Understanding DCE", O'Reilly and Associates, Inc., Sebastpal, California, 1993.

[Klas94] W. Klas, G. Fischer, K. Aberer, "Integrating Relational and Object-Oriented Database Systems using a Metaclass Concepts", Journal of Systems Integration, Volume 4, Number 4, Kluwer Academic Publisher, 1994.

[Korsh88] J. Korsh, L. Garrett, "Data Structures, Algorithms, and Program Style Using C", PWS-KENT Publishing Company, Boston, 1988.

[Krishnamurthy87] V. Krishnamurthy, Y. Su, et al., "A Distributed Database Architecture for an Integrated Manufacturing Facility", Second Symp. Knowledge-Based Integrated Info. Sys. Eng., May 1987.

[Kulkarni94] D. Kulkarni, A. Banerji, D. Cohn, "Operating System Support for Cooperation in Distributed OODBs", "Distributed Object Management, by M. Tamer Ozsu, U. Dayal, P. Valduriez", Morgan Kufmann Publishers, 1994.

[Lee97] I. Lee, H. Yeom, J. Lee, "Performance evaluation strategies for global deadlock resolution in multidatabase systems", Journal of KISS[B], Software Application, Vol. 24, Issue 5, pp. 487-500, May 1997.

[Litwin86] W. Litwin, A. Abdellatif, "Multidatabase Interoperability", IEEE Computer, pp. 10-18, December 1986.

[Litwin88] W. Litwin, "From database systems to multi-database systems: Why and How", "In Proceedings of 6th British National Conference on Databases", July 1988.

[Liu95] L. Liu, C. Pu.,"The distributed interoperable object model and its application to large-scale interoperable database systems", ACM International Conference on Information and Knowledge Management (CIKM'95), Baltemore, Maryland, USA, November 1995.

[Liu96] L. Liu, C. Pu., " An Object-oriented Approach to Interoperable Heterogeneous Information Sources", Invited Paper in Proceedings of the Seventh International Hong Kong Computer Society Database Workshop, Hong Kong (May 1996) (Springer Verlag) pp49-65.

[Luker94] P. Luker, "There's More to OOP Than Syntax", SIGCSE Bulletin, February 1994.

[Macleod93] D. Macleod, D. Fang, S. Ghandeharizadeh, A. Si, "The design, implementation, and evaluation of of an object-based sharing mechanism for federated database systems", In Proceedings of International Conference on Data Engineering, Vienna Austria, 1993.

[Marks96] D. Marks, P. Sell, B. Thuraisingham, "MOMT: A multilevel object modeling technique for designing secure database applications", Journal of Object-Oriented programming, July-August 1996, Vol. 9, No. 4, pp. 22-29.

[Miura95] T. Miura, "Optimizing complex objects requires in a visual data manipulation language", SEKE '95, 7th International Conference on Software Engineering and Knowledge Engineering, pp. 153-7, 1995.

[Monarchi92] D. Monarchi, G. Puhr, "A Research Typology for Object-Oriented Analysis and Design", Communications of the ACM, September 1992, Volume 35, Number 9.

[Mowbray95] T. Mowbray, R. Zahavi. The Essential CORBA: Systems Integration Using Distributed Objects. John Wiley, New York, 1995.

[Munakata97] K. Munakata, "Integration of heterogeneous information sources", System Control and Information, Vol. 40 Issue 12, pp. 512-21, 1997.

[Murphy94] J. Murphy, J. Grimson, "The Jupiter System: A prototype for Multi-database Interoperability", 12th British National Conference on Databases, BNCOD 12 Guildford, United Kingdom, Proceedings, Springer-Verlag, July 1994

[Murthy] V. Murthy, "Probabilistic Security Protocol for Real-Time Authentication in Distributed Databases", menu script.

[Muth91] P. Muth, T. Rakow, "Atomic Commitment for Integrated Database Systems", IEEE, 1991

[Narasimhan97] P. Narasimhan, L. Moser, P. Melliar-Smith, "Exploring the Enternet Inter-ORB Protocol Interface to Provide CORBA with Fault Tolerance", Proceedings of the 3rd USENIX Conference on Object-Oriented Technologies and Systems COOTS, pp. 81-90, 1997.

[Needham78] R. Needham, R. Schroeder, "Using encryption for authentication in large network of computers", Comm. ACM. 21, December 1978, pp. 933-999.

[OMG91] Object Management Group and X/Open. The common object request broker Architecture and specification. Technical Report OMG Document No. 91.12.1, Object Management Group and X/Open, Framingham, Massachusetts, 1991.

[OSF92] Open Systems Foundation, The OSF distributed computing environment. Technical Report OSF-DCE-PD-1090-4, Open Systems Foundation, Cambridge, Massachusetts, 1992.

[OSPF99] OSPF and the Internet, URL: http://www.livingston.com/marketing/whitepapers/ospf_whitepaper.html, Lucent Technology, 1999.

[Ozsu91] M. Ozsu, P. Valduriez, "Principles of Distributed Database Systems", "Prentice-Hall", 1991.

[Ozsu91] M. Ozsu, P. Valduriez, "Principles of Distributed Database Systems", "Prentice-Hall", 1991.

[Ozsu93] M. Ozsu, U. Dayal, P. Valduriez, "Distributed Object Management", Morgan Kaufmann, 1993.

[Ozsu93a] M. Ozsu, "Interoperability and Multi-database Systems", "The Relational Journal", April/May 1993.

[Papakonstantinou95] Y. Papakonstantinou, H. Garcia-Molina, J. Windom, "Object Exchange Across Heterogeneous Information Sources", In Proceedings of International Conference on Data Engineering, Taiwan, March 1995.

[Peters88] L. Peters, "Advanced Structured Analysis and Design", Printice Hall, 1988.

[Pozefsky96] D. Pozefsky, R. Turner, A. K. Edwards, S. Sarkar, J. Mathew, G. Bollella, K. Tracey, D. Poirier, J. Fetvedt, W. S. Hobgood, W. A. Doeringer, D. Dykeman, "Multiprotocol Transport Networking: Eliminating application dependencies on communications protocols", "IBM Systems Journal, Network Technologies and Systems", Volume 34, Number 3, pp.472 – 500, 1996.

[Prabhu96] M. Prabhu, S. Raghavan, "Security in computer networks and distributed systems", *IEEE Computer Communications, Issue 19, 1996, pp. 379-388.*

[Qian95] X. Qian, L. Raschid, "Query Interoperation among Object-Oriented and Relational Databases", Proceedings of the International Conference on Data Engineering, 1995.

[Raschid94] L. Raschid, "Issues in Supporting Interoperable Query Processing with Multiple Heterogeneous Information Servers", Workshop on the Information Technology and systems, International Conference on Information Systems, December 1994.

[Reinwald96] B. Reinwald, T. Lehman, H. Pirahesh, V. Gottemukkala, "Storing and using objects in a relational database", IBM Systems Journal, Vol. 35, No. 2, 1996, pp172-191.

[Rumbaugh91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, "Object-Oriented Modeling and Design", Engwood Cliffs, New Jersey, Printice Hall 1991.

[Rumbaugh99] J. Rumbaugh, I. Jacobson, G. Booch The Unified Modeling Language Reference Manual", Addison Wesley, 1999.

[Shan93] M. Shan, "Pegasus architecture and design principles", In Proceedings of ACM/SIGMOD Annual Conference on Management of Data", 1993.

[Sheldon94] T. Sheldon, "LAN Times Encyclopaedia of Networking", Osborne McGraw-Hill, 1994.

[Sheth90] A. Sheth, J. Larson, "Federated database systems and manageing distributed, heterogeneous, and autonomous databases", ACM Computing Surveys, Vol. 22, No. 3, pp. 183-226, September 1990.

[Siberschartz94] A. Siberschartz, M. Stonebraker, J. Ulman, "Database Systems: Achievements and opportunities", Readings in database systems, Murgan Kufmann Publisher, Sanfrancisco, California, 1994.

[Silberschatz97] A. Silberschatz, H. Korth, S. Sudarshan, "Database System Concepts", McGraw-Hill, 1997.

[Simon95] A. Simon, "Strategic database technology: Management for the year 2000", Murgan Kaufmann Publishers, San Francisco, California, 1995.

[Siyan94] K. Siyan, "CNE Training Guide, Netware TCP/IP and NFS", New Riders Publishing, Indianapolis, Indiana, 1994.

[Smith81] J. Smith, "Multibase - Integrating Heterogeneous Distributed Database Systems", AFIPS Conf. Proc., Vol 50, 1981, pp. 487-499.

[Smith81] J. Smith, P. Bernstein, et al., "Multibase—Integrating Heterogeneous Distributed Database Systems", Proceedings of AFIPS, pp 287-499, 1981.

[Staniszkis] W. Staniszkis, "Integrating Heterogeneous Databases", "State of the Art Report on Relational Database", Pergamon press Ltd., Chapter 16, pp 229-287.

[Steiner88] J. Steiner, C. Neuman, J. Schiller, Kerberos, "An authentication service for open network systems", Project Athena, Technical report, MIT 1988.

[Stewart90] T. Stewart, "SIOIS—Standard interfaces or interface standards", *Proceedings IFIP INTERACT'90* (Cambridge, U.K., 27–31 August), 1990.

[Stonebraker90] M. Stonebraker, A. Rowe, B. Lindsay, J. Gray, M. Carey, M. Brodie, P. Bernstein, D. Beech, 1990, Third-generation Database System Manifesto", SIGMOD Record, 19(3), September 1990.

[Stonebraker94] M. Stonebraker, "Distributed database systems", Readings in database systems, Morgan Kufmann Publisher, pp. 507-514, 1994.

[Subrahamanian94] S. Subrahamanian, "Amalgamating Knowledge Bases", "ACM Trans. on Database Systems", June 1994.

[Tanenbaum87] A. Tanenbaum, "Operating Systems, design and implementation", "Printice-Hall", 1987.

[Templeton87] M. Templeton, D. Brill, et al., "Mermaid—A Front-End to Distributed Heterogeneous Databases", Proceedings of IEEE, pp 695-708, No. 5, May 1987.

[Terplan87] K. Terplan, "Communication networks management", Printice-Hall International, 1987.

[Ting95] T. Ting, " How secure is secure: Some thoughts on security metrics", Database security IX status and prospects, Proceedings of the 9th annual IFIP TC11 working conference on database security, August 1995.

[Tomasic96] A. Tomasic, L. Raschid, P. Valduriez, "Scaling Heterogeneous Databases and Design of Disco", Proceedings of the International Conference on Distributed Computer Systems, 1996.

[Verma95] R. Verma, "Unique normal forms and confluence of rewrite systems : Persistence", IJCAI-95, Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 362-8, Vol. 1, 1995.

[W3C98] W3C Extensible Markup Language (XML) 1.0 Recommendation, REC-xml-19980210, URL: http://www.w3.org/TR/1998/REC-xml-19980210, 10 February 1998.

[Weinberg92] R. Weinberg, T. Guimaraes, R. Heath, "An Introduction to Object Orientation", Data Resources Management, Winter 1992.

[Wiederhold88] G. Wiederhold, "File Organization for Database Design", McGRAW-HILL International Edition, 1988.

[Wiederhold92] G. Wiederhold, "Mediators in the architecture of future information systems", IEEE Computer Magazine, March 1992.

[Wiederhold93] G. Wiederhold, "Intelligent integration of information", In Proceedings of ACM/SIGMOD Annual Conference on Management of Data", 1993.

Conference paper in the IPSJ, 1990.

[Yourdon83] E. Yourdon, "Structured Systems Analysis and Design", Printice Hall, 1983.

[Yourdon93] E. Yourdon, "Yourdon Systems Method, Model-Driven System Development", Printice Hall, 1993.

Appendix A

The IE UML classes

This appendix consists of two complementary parts. The first is the UML version of the various interrelated classes. The second is a spreadsheet contains the movement parts from the local IE knowledge to the global IE knowledge.

Figure A.1: Local Interoperation Engine classes.
Figure A.2: Global Interoperation Engine classes.
Figure A.3: Local Interoperation Engine to Global Interoperation Engine data transfers.
Figure A.4: Detailed Local Interoperation Engine classes.
Figure A.5: Local Users and Systems profile classes.
Figure A.6: Detailed Global Interoperation Engine classes.
Figure A.7: Global Users and Systems profile classes.

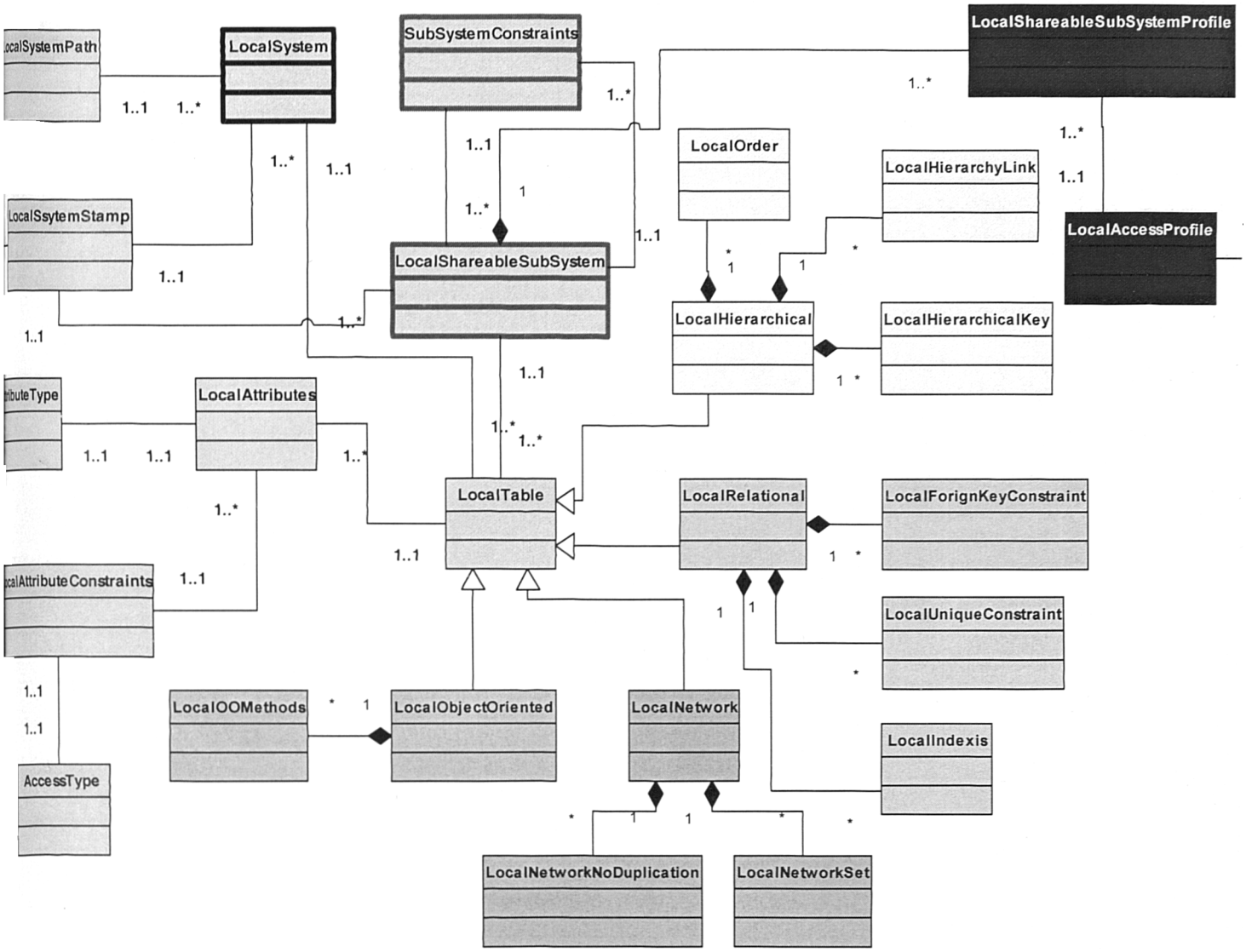

LocalSystemPath
LocalSystem
SubSystemConstraints
LocalShareableSubSystemProfile
LocalSsytemStamp
LocalShareableSubSystem
LocalOrder
LocalHierarchyLink
LocalAccessProfile
LocalHierarchical
LocalHierarchicalKey
tributeType
LocalAttributes
LocalTable
LocalRelational
LocalForignKeyConstraint
localAttributeConstraints
LocalUniqueConstraint
LocalOOMethods
LocalObjectOriented
LocalNetwork
LocalIndexis
AccessType
LocalNetworkNoDuplication
LocalNetworkSet
1..1
1..*
1

Figure A.1

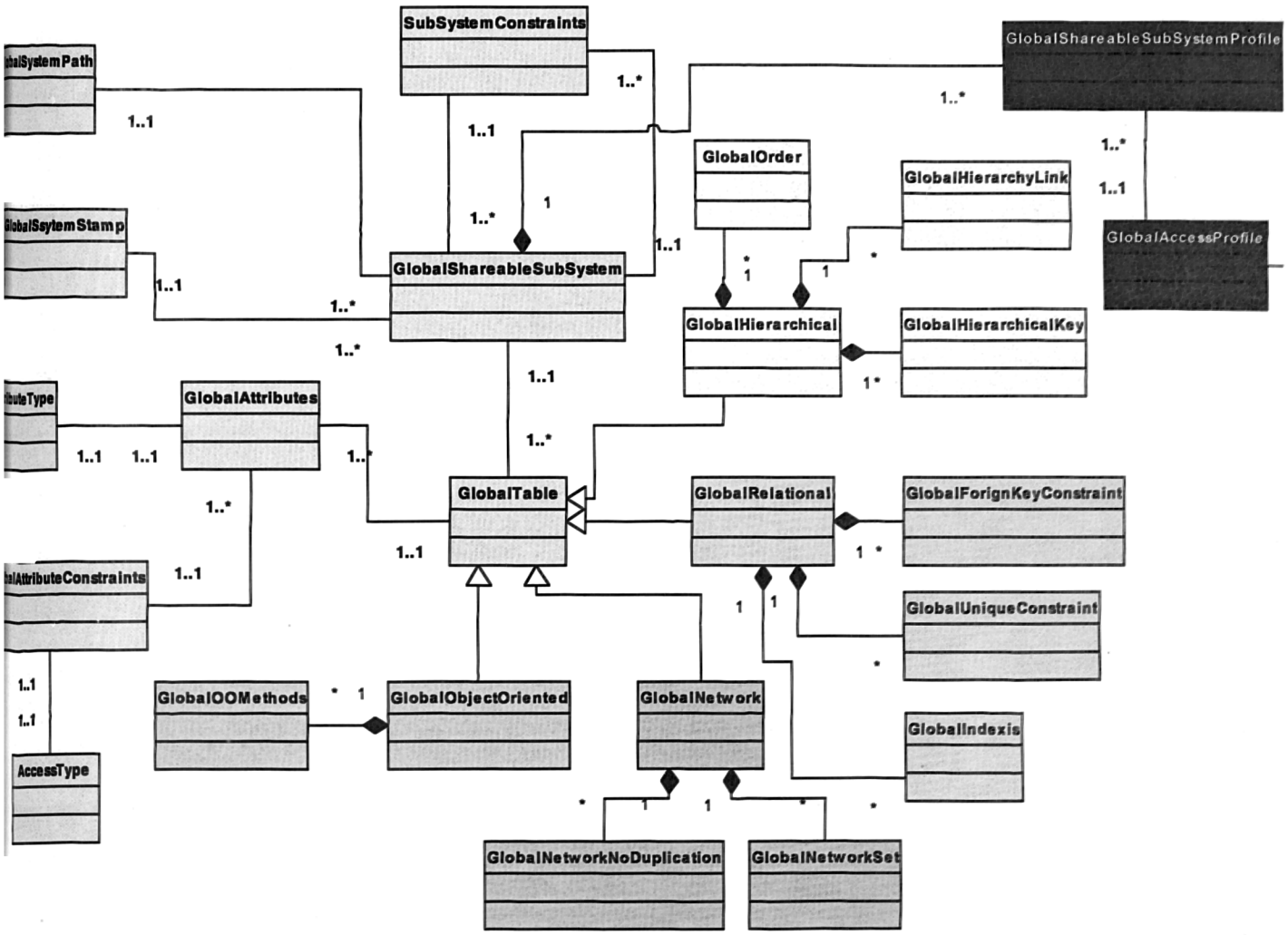

SubSystemConstraints
GlobalShareableSubSystemProfile
GlobalOrder
GlobalHierarchyLink
GlobalAccessProfile
GlobalShareableSubSystem
GlobalHierarchical
GlobalHierarchicalKey
GlobalAttributes
GlobalTable
GlobalRelational
GlobalForignKeyConstraint
GlobalUniqueConstraint
GlobalIndexis
GlobalOOMethods
GlobalObjectOriented
GlobalNetwork
AccessType
GlobalNetworkNoDuplication
GlobalNetworkSet

Figure A.2

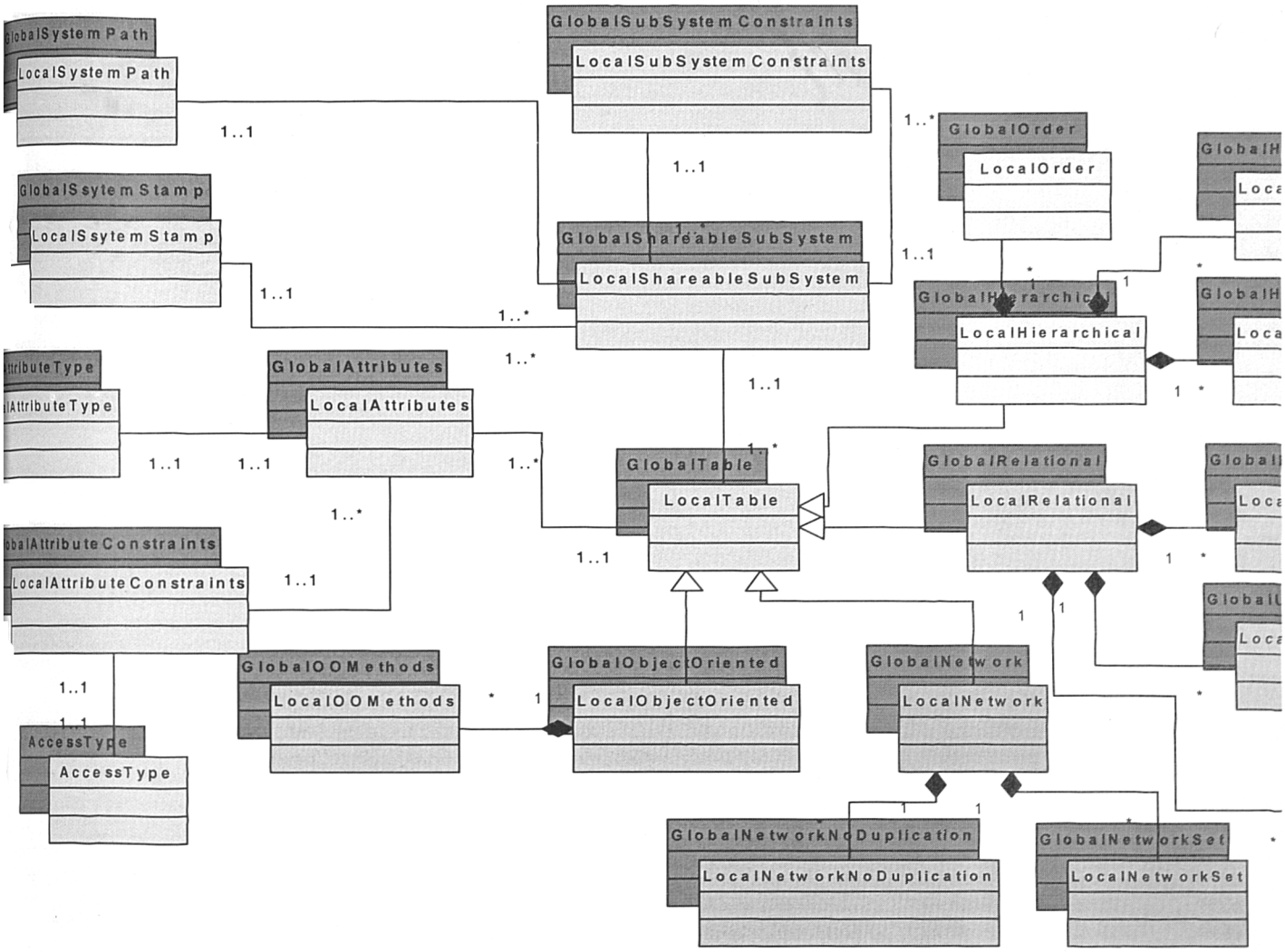

GlobalSystemPath
LocalSystemPath
GlobalSubSystemConstraints
LocalSubSystemConstraints
GlobalOrder
LocalOrder
GlobalSsytemStamp
LocalSsytemStamp
GlobalShareableSubSystem
LocalShareableSubSystem
GlobalHierarchical
LocalHierarchical
GlobalAttributes
LocalAttributes
GlobalTable
LocalTable
GlobalRelational
LocalRelational
LocalAttributeConstraints
GlobalOOMethods
LocalOOMethods
GlobalObjectOriented
LocalObjectOriented
GlobalNetwork
LocalNetwork
AccessType
GlobalNetworkNoDuplication
LocalNetworkNoDuplication
GlobalNetworkSet
LocalNetworkSet
1..1
1..*
*
1

Figure A.3

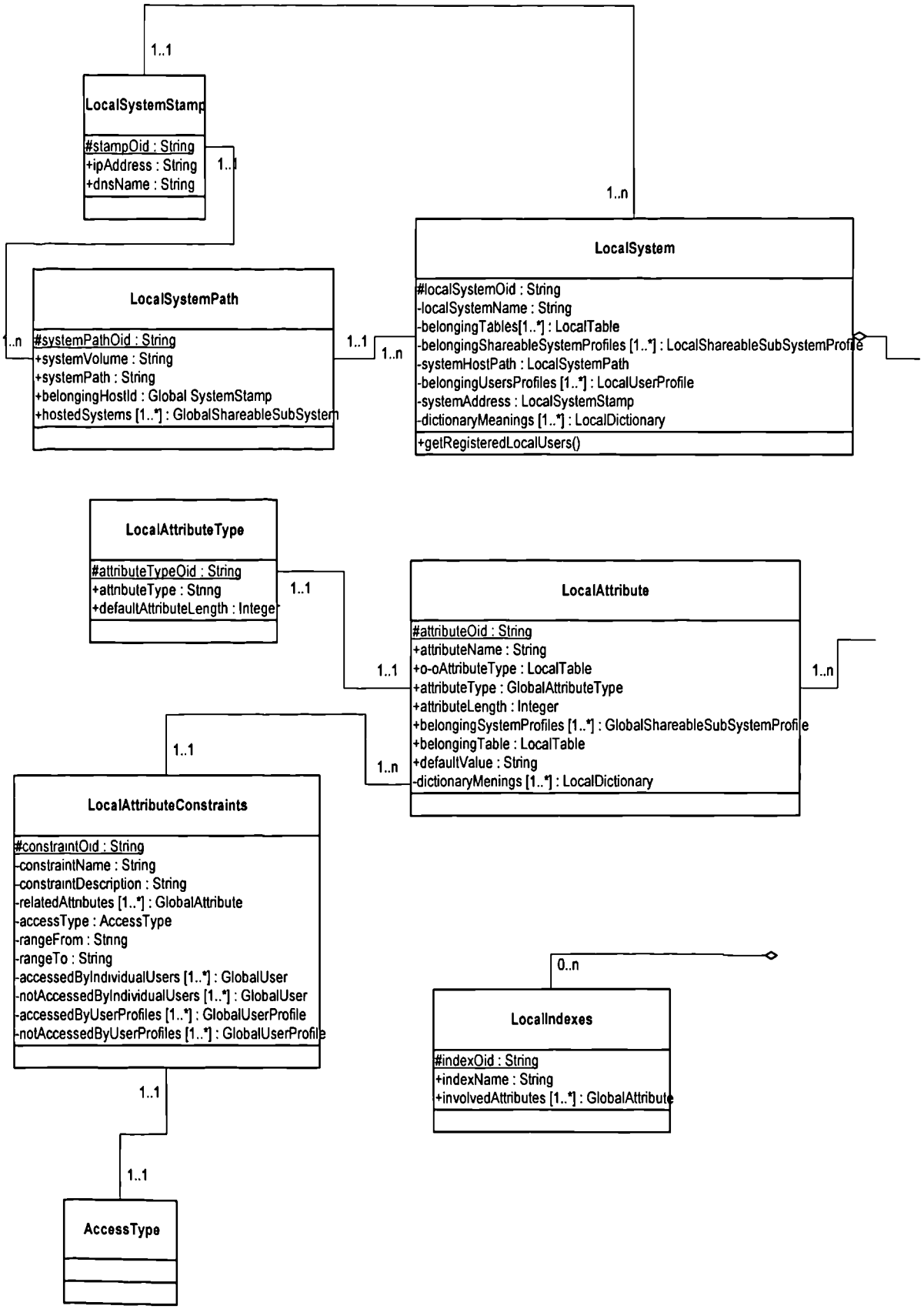


Figure A.4 (Part 1)

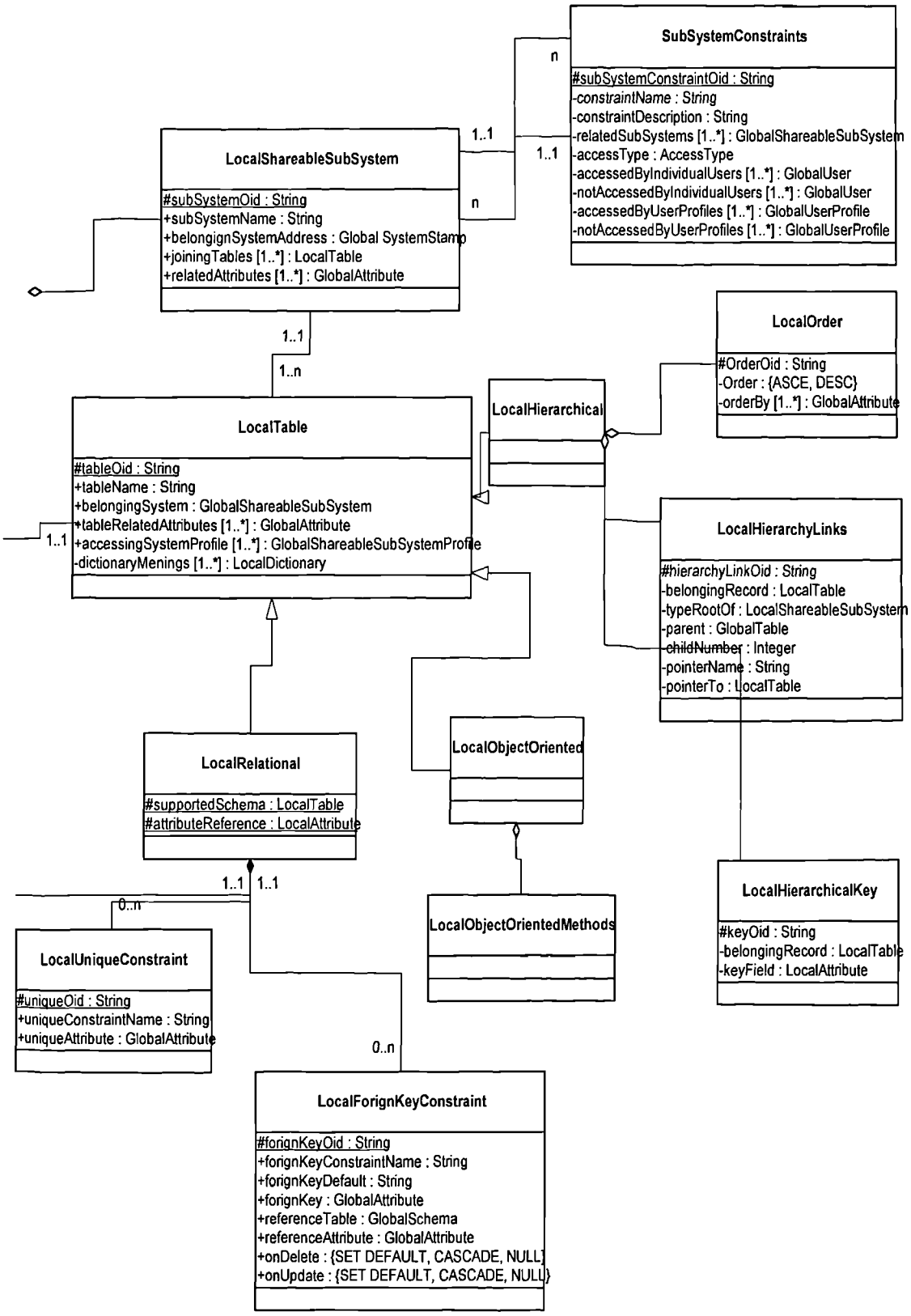


Figure A.4 (Part 2)

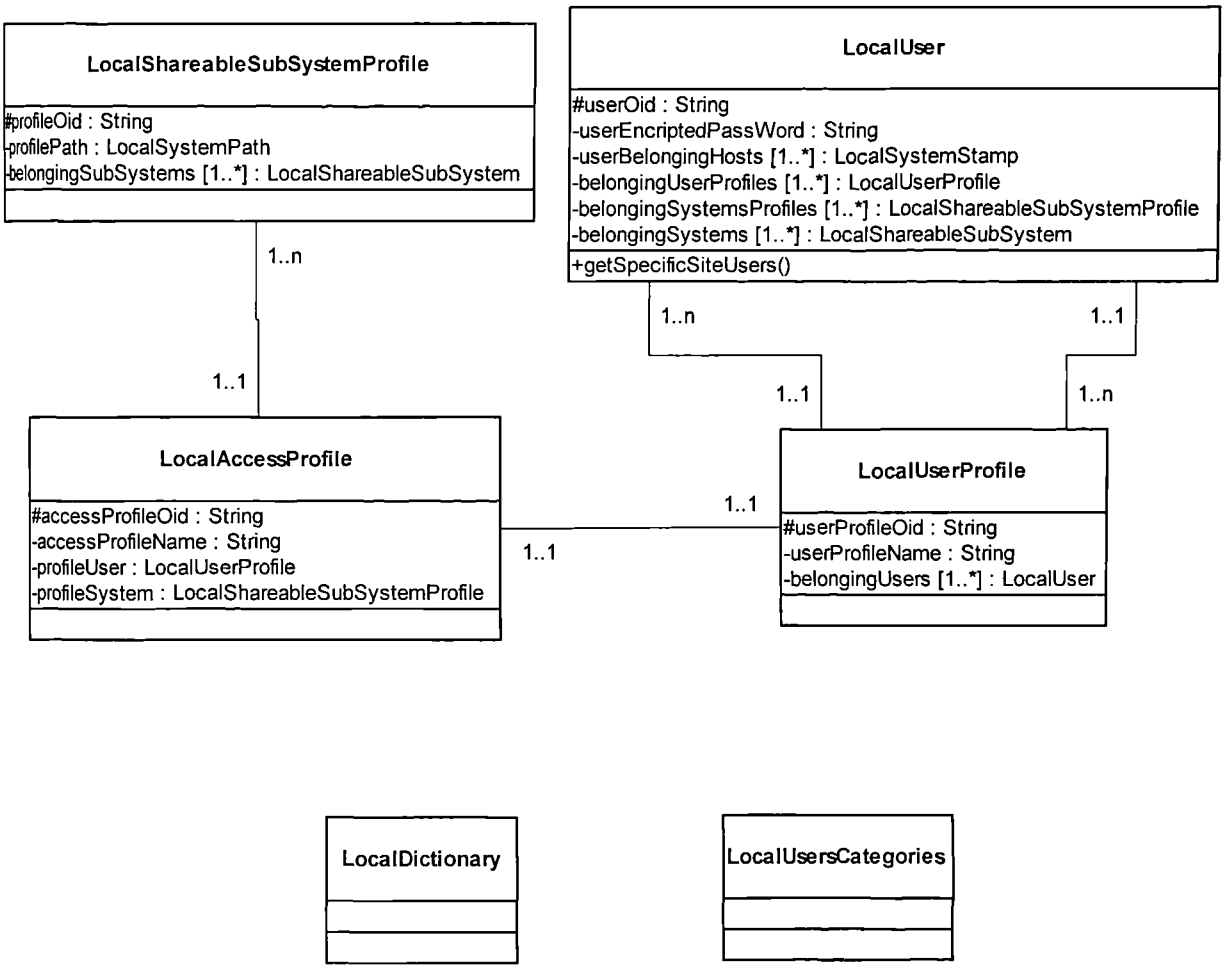

LocalShareableSubSystemProfile
#profileOid : String
-profilePath : LocalSystemPath
-belongingSubSystems [1..*] : LocalShareableSubSystem
LocalUser
#userOid : String
-userEncriptedPassWord : String
-userBelongingHosts [1..*] : LocalSystemStamp
-belongingUserProfiles [1..*] : LocalUserProfile
-belongingSystemsProfiles [1..*] : LocalShareableSubSystemProfile
-belongingSystems [1..*] : LocalShareableSubSystem
+getSpecificSiteUsers()
1..n
1..1
1..n
1..1
1..1
1..n
LocalAccessProfile
#accessProfileOid : String
-accessProfileName : String
-profileUser : LocalUserProfile
-profileSystem : LocalShareableSubSystemProfile
1..1
1..1
LocalUserProfile
#userProfileOid : String
-userProfileName : String
-belongingUsers [1..*] : LocalUser
LocalDictionary
LocalUsersCategories

Figure A.5

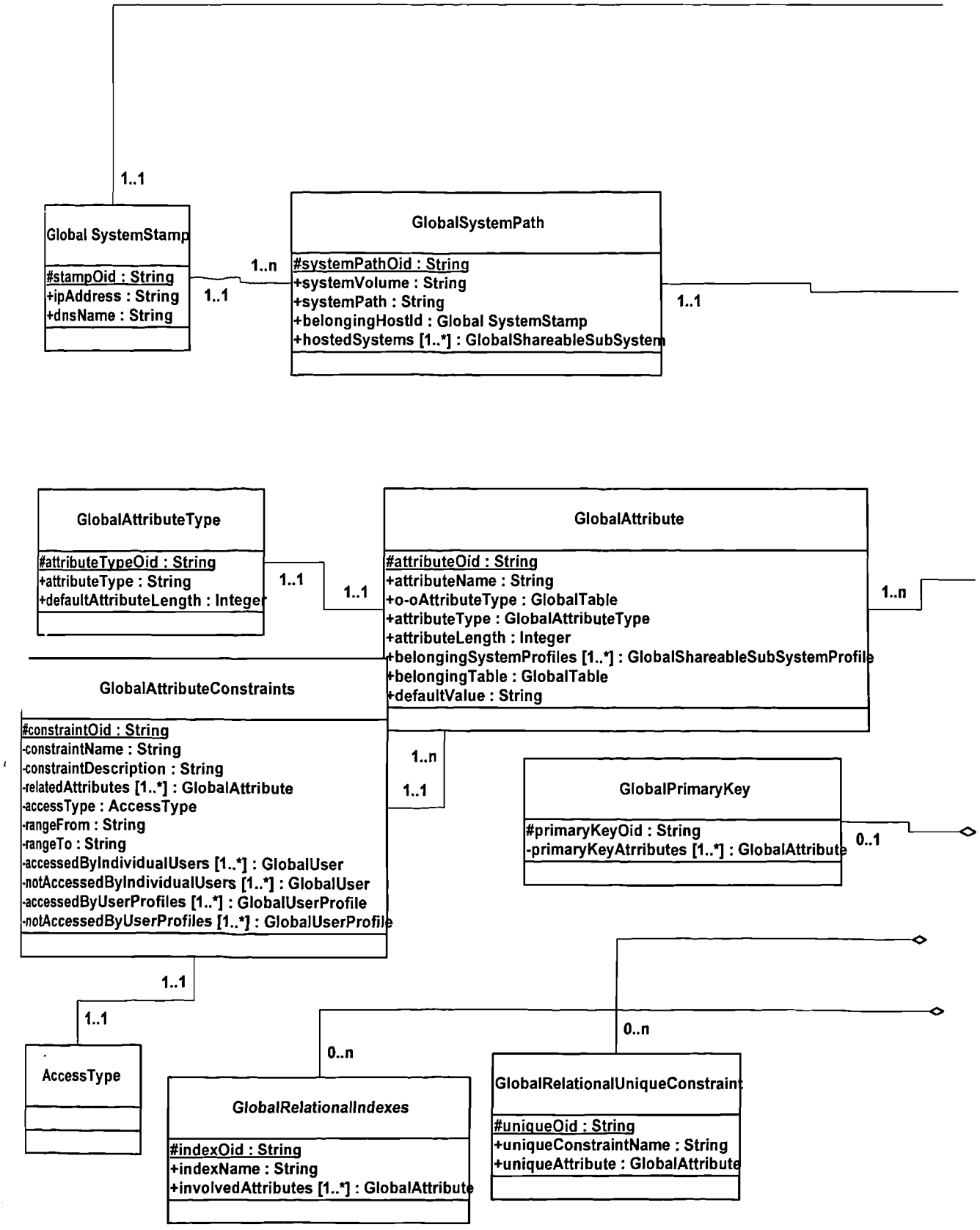


Figure A.6 (Part 1)

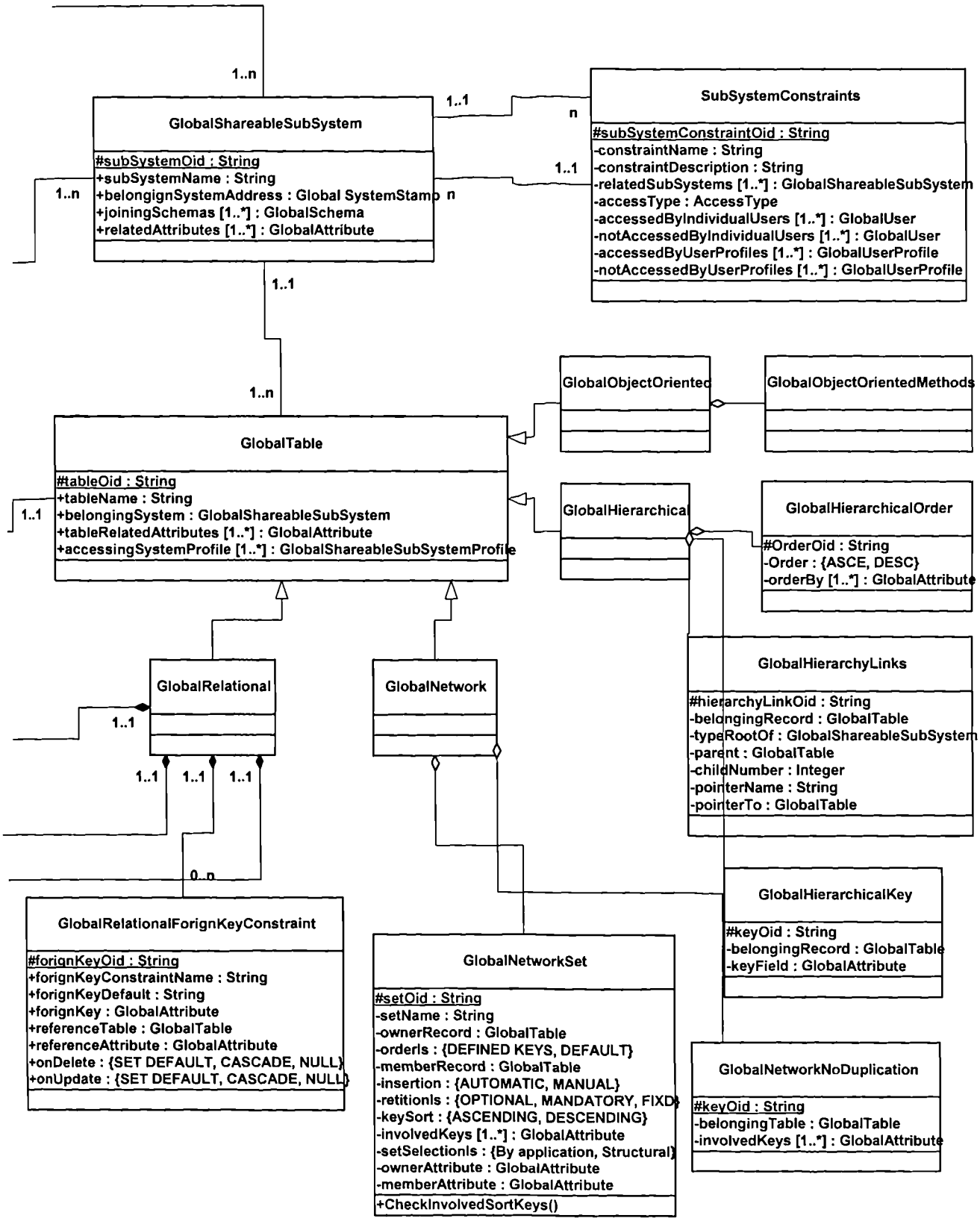


Figure A.6 (Part 2)

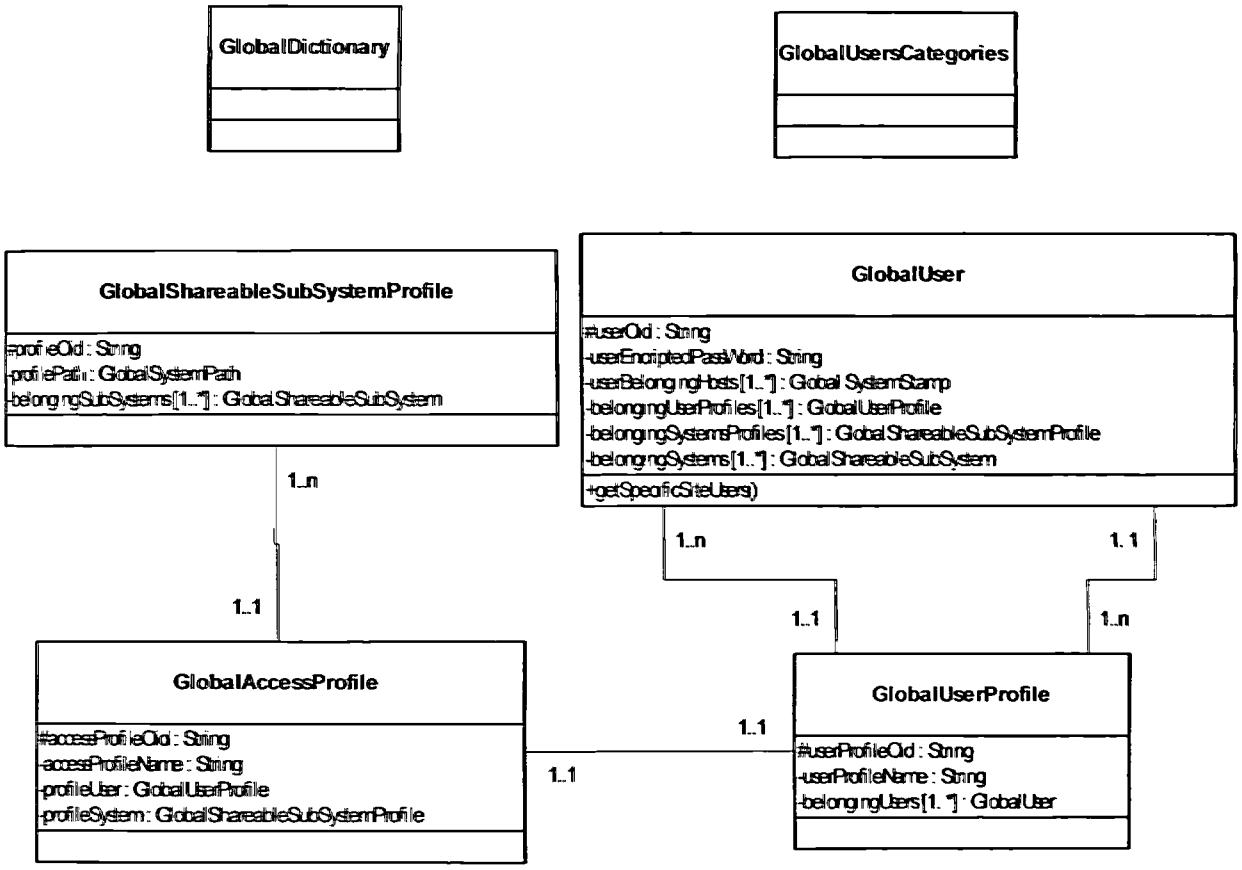

GlobalDictionary
GlobalUsersCategories
GlobalShareableSubSystemProfile
#profileOid : String
-profilePath : GlobalSystemPath
-belongingSubSystems[1..*] : GlobalShareableSubSystem
GlobalUser
#userOid : String
-userEncriptedPassWord : String
-userBelongingHosts[1..*] : GlobalSystemStamp
-belongingUserProfiles[1..*] : GlobalUserProfile
-belongingSystemsProfiles[1..*] : GlobalShareableSubSystemProfile
-belongingSystems[1..*] : GlobalShareableSubSystem
+getSpecificSiteUsers()
1..n
1..1
1..n
1..1
1..1
1..n
GlobalAccessProfile
#accessProfileOid : String
-accessProfileName : String
-profileUser : GlobalUserProfile
-profileSystem : GlobalShareableSubSystemProfile
1..1
1..1
GlobalUserProfile
#userProfileOid : String
-userProfileName : String
-belongingUsers[1..*] : GlobalUser

Figure A.7

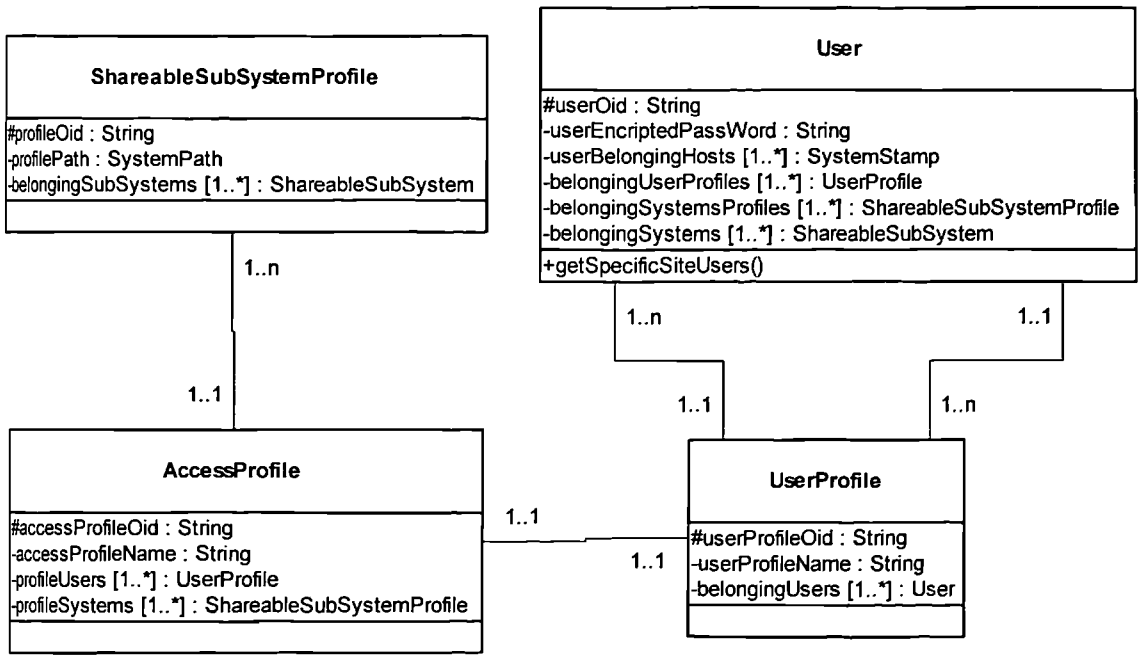

ShareableSubSystemProfile
#profileOid : String
-profilePath : SystemPath
-belongingSubSystems [1..*] : ShareableSubSystem
User
#userOid : String
-userEncriptedPassWord : String
-userBelongingHosts [1..*] : SystemStamp
-belongingUserProfiles [1..*] : UserProfile
-belongingSystemsProfiles [1..*] : ShareableSubSystemProfile
-belongingSystems [1..*] : ShareableSubSystem
+getSpecificSiteUsers()
1..n
1..1
1..n
1..1
1..n
1..1
AccessProfile
#accessProfileOid : String
-accessProfileName : String
-profileUsers [1..*] : UserProfile
-profileSystems [1..*] : ShareableSubSystemProfile
1..1
1..1
UserProfile
#userProfileOid : String
-userProfileName : String
-belongingUsers [1..*] : User

Figure A.8

The following pages spreadsheet shows the transmitted data from the local Interoperation Engine to the global Interoperation Engine knowledgebase.

n
ated, not displayed

Class Name	Attribute	Type	Global IE Class Name	Att
mStamp	stampOid	String	GlobalSystemStamp	stam
	ipAddress	String		ipAd
	dnsName	String		dnsI
mPath	systemPathOid	String	GlobalSystemPath	syst
	systemVolume	String		syst
	systemPath	String		syst
	belongingHostId	Local SystemStamp		belo
	hostedSystems [1..*]	LocalShareableSubSystem		host
bleSubSystem	subSystemOid	String	GlobalShareableSubSystem	subS
	subSystemName	String		subS
	belongignSystemAddress	Local SystemStamp		belo
	joiningSchemas [1..*]	LocalSchema		joini
	relatedAttributes [1..*]	LocalAttribute		relat
ute	attributeOid	String	GlobalAttribute	attri
	attributeName	String		attri
	o-oAttributeType	LocalTable		o-oA
	attributeType	LocalAttributeType		attri
	attributeLength	Integer		attri
	belongingSystemProfiles [1..*]	LocalShareableSubSystemProfile		belo
	belongingTable	LocalTable		belo
	defaultValue	String		defa
uteType	attributeTypeOid	String	GlobalAttributeType	attri
	attributeType	String		attri
	defaultAttributeLength	Integer		defa
uteConstraints	constraintOid	String	GlobalAttributeConstraints	cons
	constraintName	String		cons
	constraintDescription	String		cons
	relatedAttributes [1..*]	LocalAttribute		relat
	accessType	AccessType		acce
	rangeFrom	String		rang
	rangeTo	String		rang
	accessedByIndividualUsers [1..*]	LocalUser		acce
	notAccessedByIndividualUsers [1..*]	LocalUser		notA
	accessedByUserProfiles [1..*]	LocalUserProfile		acce
	notAccessedByUserProfiles [1..*]	LocalUserProfile		notA
	tableOid	String	GlobalTable	table
	tableName	String		table
	belongingSystem	LocalShareableSubSystem		belo
	tableRelatedAttributes [1..*]	LocalAttribute		table
	accessingSystemProfile [1..*]	LocalShareableSubSystemProfile		acce

			GlobalRelational	
lIndexes	indexOid	String	GlobalRelationalIndexes	indexO
	indexName	String		indexN
	involvedAttributes [1..*]	LocalAttribute		involv
lUniqueConstrain	uniqueOid	String	GlobalRelationalUniqueConstrain	uniqu
	uniqueConstraintName	String		uniqu
	uniqueAttribute	LocalAttribute		uniqu
lForignKeyConstrain	forignKeyOid	String	GlobalRelationalForignKeyConstrain	forign
	forignKeyConstraintName	String		forign
	forignKeyDefault	String		forign
	forignKey	LocalAttribute		forign
	referenceTable	LocalSchema		refere
	referenceAttribute	LocalAttribute		refere
	onDelete	{SET DEFAULT, CASCADE, NULL}		onDel
	onUpdate	{SET DEFAULT, CASCADE, NULL}		onUp
rk			GlobalNetwork	
rkSet	setOid	String	GlobalNetworkSet	setOi
	setName	String		setNa
	ownerRecord	LocalTable		owner
	orderIs	{DEFINED KEYS, DEFAULT}		orderI
	memberRecord	LocalTable		membe
	insertion	{AUTOMATIC, MANUAL}		insert
	retitionIs	{OPTIONAL, MANDATORY, FIXD}		retitio
	keySort	{ASCENDING, DESCENDING}		keySo
	involvedKeys [1..*]	LocalAttribute		involv
	setSelectionIs	{By application, Structural}		setSe
	ownerAttribute	LocalAttribute		owner
	memberAttribute	LocalAttribute		membe
NoDuplication	keyOid	String	GlobalNetworkNoDuplicatior	keyOi
	belongingTable	LocalTable		belon
	involvedKeys [1..*]	LocalAttribute		involv

			GlobalHierarchical	
alOrder	OrderOid	String	GlobalHierarchicalOrder	Order
	Order	*{ASCE, DESC}*		Order
	orderBy [1..*]	LocalAttribute		orderB
yLinks	hierarchyLinkOid	String	GlobalHierarchyLinks	hierarc
	belongingRecord	LocalTable		belong
	typeRootOf	LocalShareableSubSystem		typeRo
	parent	LocalTable		parent
	childNumber	Integer		childN
	pointerName	String		pointe
	pointerTo	LocalTable		pointe
alKey	keyOid	String	GlobalHierarchicalKey	keyOic
	belongingRecord	LocalTable		belong
	keyField	LocalAttribute		keyFie
Oriented			GlobalObjectOriented	
OrientedMethods			GlobalObjectOrientedMethods	
	userOid	String	GlobalUser	userO
	userEncriptedPassWord	String		userEr
	userBelongingHosts [1..*]	Local SystemStamp		userBe
	belongingUserProfiles [1..*]	LocalUserProfile		belong
	belongingSystemsProfiles [1..*]	LocalShareableSubSystemProfile		belong
	belongingSystems [1..*]	LocalShareableSubSystem		belong
file	userProfileOid	String	GlobalUserProfile	userPr
	userProfileName	String		userPr
	belongingUsers [1..*]	LocalUser		belong
eSubSystemProfile	profileOid	String	GlobalShareableSubSystemProfile	profile
	profilePath	LocalSystemPath		profile
	belongingSubSystems [1..*]	LocalShareableSubSystem		belong
Profile	accessProfileOid		GlobalAccessProfile	access
	accessProfileName			access
	profileUser	LocalUserProfile		profile
	profileSystem	LocalShareableSubSystemProfile		profile

mConstraints	subSystemConstraintOid	String	**SubSystemConstraints**	subS
	constraintName	String		const
	constraintDescription	String		const
	relatedSubSystems [1..*]	**LocalShareableSubSystem**		relate
	accessType	**AccessType**		acce
	accessedByIndividualUsers [1..*]	**LocalUser**		acce
	notAccessedByIndividualUsers [1..*]	**LocalUser**		notAc
	accessedByUserProfiles [1..*]	**LocalUserProfile**		acce
	notAccessedByUserProfiles [1..*]	**LocalUserProfile**		notAc

Appendix B

The Heterogeneous Communication Protocols Layer

Abstract
The ways in which people communicate are unstopping techniques and far-reaching changes since Bell's invention of the telephone in 1876. Today, there is a huge array of new telecommunication technologies, new facilities and services, and new communication options containing voice, data, text, image, and real-time video. In the field of computer communications, a protocol is a procedure executed by two cooperating processes in order to attain a meaningful exchange of information. Of course, if the case is heterogeneous communication protocols than there should exist some sort of translation mechanism, to make the process of talking between such heterogeneous protocols a meaningful operation. This part of the work discuss issues related to communication protocols and heterogeneity of them, and propose and design a method fits with all the requirements of a heterogeneous databases in a heterogeneous environments.

B.1 Introduction

In order for two computers to talk to each other, they first need to notify each other that they are able to communicate and they can understand each other. Second, once they communicate, they must provide a method, which keeps both devices aware of the ongoing transmissions.

Protocols are collection of, or a set of rules defined in between two parts to guarantee some level of mutual understanding when both needs to talk to each other. Communication protocols comprise three main areas: 1) the method in which data is represented and coded; 2) the method in which codes are transmitted and received, and; 3) non-standard information exchanges by which two devices establish control, detect failures or errors, and initiate corrective action. Those three areas mainly govern interoperability of communication protocols. Whenever different protocols are able to talk and understand each other, than it can be said, that there is communication protocols interoperability [Ozsu91].

Interoperability of data exchange is not only transferring data across networks, but it is also transforming of true-time voice and video frames over the communication lines. This type of data requires high bandwidth, as well as, fast media machines. Although security (accuracy) of the transferred data is an important issue, It will be discussed in a subsequent chapter.

The Internet, which is the largest network in the world, has been considered as the largest example in using multi communication protocols. The *Internet* overall the world is divided into domains, or autonomous systems. A domain is a collection of hosts and routers that use the same routing protocol and are administrated by a single authority. The autonomous system is also divided into another sub-areas. This technique is introduced to put a boundary on the explosion of network updates.

Protocols running overall the world can be categorized, in terms of exchanging data, into three types. The first is the exchange happens within a single segment or network and does not required routing capabilities. The second types are those considered as interior protocols running within the autonomous system. Those are used to carry packets between routers fall in the same domain or area. The third types are the exterior routing protocols, which are mainly running in between autonomous systems. They provide a way for two neighboring

routers located at edges of their respective domains to exchange messages and information. Also with reference to framing, protocols may be subdivided into three principal groups. These are the character-oriented protocols, the byte oriented protocols, and the bit oriented protocols [Browne93].

B.2 Carrier services approaches

Carriers are companies providing telephone and data communication services in local geographic areas or between local geographic areas. They form the business unit of the national and international telecommunication network. Carriers owns and operate facilities such as switching or data communication connections, maintenance equipment, and other transmission facilities that provide communication pathways using either guided medians such as copper wires and fiber-optic cables or unguided median such as radio-waves. So carriers can be either local exchanges or long distance carriers.

Services for carrying packets in LAN-to-LAN, either from point to a single point, or from point to multiple points can be categorized in four different types. The transformation of data can be either analog or digital used by circuit switching hardware. The first type is preferred when occasional traffic between two points is required, while the second is preferred for periodic connections between numbers of different points. The third category of switching services is the dedicated line, which provide a permanent connection between two points on a leased, month-to-month bases, normally with an initial setup charge. Lines are available in speed ranging from 56 Kbits/sec to 45 Mbits/sec they are suitable for handling constant traffic between two points. The last category, which is the most flexible technique, is the packet switching services. This category provides transformation of data in varying amounts. They provide simultaneous connections to many points and bandwidth on demand. The next section explains each of the previous categories concentrating on the design of each, and the possibility of merging these technologies for the purpose of gaining complete carrier services interoperability [Sheldon94].

Carrier services can be considered as an airplane. Whenever airplane can flight from point to point in minimal time, then it will be the best to the passenger, who is the user of the facility. Also, the number of passengers that can be carried at one time is another important factor, reflecting the reliability of the service. Most of the passengers requesting that service will reach the other point at the best and minimal time.

When generally considering such protocols used for handling data through communication lines of distributed networks, it is really thinking of huge amount of policies and roles where each of those has certain amount of limitations. Performance of communication protocols is dependent on the quality of the carrier services they are working under them. The faster technology the carrier service can offer with security facilities plus scaleable bandwidth offers the better carrying service and transfer can be reliable. For example, fast packet switching services currently used such as asynchronous transfer mode (ATM) and frame relay (FR) provide unique services that can scale up as user requirements grow. ATM is shown to have special significance both for today's multi protocol networks and tomorrow's multimedia networks.

Carrier services can be categorized to number of different services such as the traditional switched analogy lines, circuit switched services, dedicated digital services, and the new promising technology, which is the packet switching services. All of these services are suppose to cooperate and be compatible with each other to achieve the interoperability of communication protocols. The protocol of high throughput and bandwidth in the heterogeneous communication protocols scenario will be degraded by low services. In that the high bandwidth protocols should accommodate the low bandwidth protocols when communication is required between them [Hess96].

Carrier services have been gradually improved since the first analog carrying services invented. These days, carries can offer very fast technologies with support to very heavy load of data transfer rates. These new technologies are called packet switching services. A packet switching networks are mesh topologies of interconnections provided by carrier services

through which packets travel from source to destination, where every packet is routed through the network as a self-contained entity. As an example, the X.25 is a standard, and often revised protocol that has been a workhorse packet switching service since 1976. It is suitable for light loads and mainly used for mainframe systems. It is not suitable for LAN-to-LAN traffic because it is slow and requires a large portion of the bandwidth to handle error checking. This was important with the low-quality analog telephone lines, but is not needed today. Frame relay comes after the X.25 and considered as enhancement for it. It is relatively error free and does not require the extensive error checking applied by the first. Frame relay is an excellent choice for any to any topologies. Frame relay does not have the capabilities to transmit and support many types of traffic such as voice, data, real-time video, CD-quality audio and imaging. In that cell switching networks, namely Asynchronous Transfer Mode, provide unique services that can scale up as user requirements grow. This technology has the potential to revolutionize the way computer networks are built. It is viable for both local and wide area networks. High speed implementations of the ATM which ranging from 155 Mbit/sec to 622 Mbit/sec make the technology to be the first to be used in the world wide interoperability of databases of any type [Terplan87].

These days the Internet is working towards running real multimedia applications. Multimedia database systems would most likely interact with higher-level protocols. To satisfy bandwidth requirements, delay, bit error requirements, and data streams carrying multimedia information will need to be sent over flows having performance guarantees. Also multimedia facilities need abstractions hide the complexities of resource allocation, network access control, and session management from the application level. Future multimedia will require very fast carrying service such as the ATM technology.

ATM is a broadband fast transport technology, based on fixed length cells expected be widely used for carrying multimedia traffic. ATM is capable of handling wide area networks as well as local area networks. Unlike other transport technologies such as frame relay and LAN systems in which packets can vary in size, all cells in ATM are the same size. In that it is very easy to predict the time the packets can transmit from one side to another side in the network by using this technology. The reason behind this is that, variable length packets can cause traffic delays at switches; in the same way cars must wait for long tracks to make turns at busy intersections [Browne93].

B.3 The communication protocols

Each layer in the Open System Interconnectivity OSI model has got number of specific protocols handling all the issues of the layer itself, as well as the communication of that layer with the upper and lower adjacent layers. The OSI model, which is the international open system standard, has defined number of roles to be followed by vendors to guarantee mutual interaction between different vendors' products. The OSI seven layers are the application, presentation, session, transport, network, data-link, and physical layer. That layer is discussed individually in the next section. In terms of function they are grouped into three categories. They are the application-level network service users, the transport services, and the network services.

Application-level network service users encompass the application, presentation, and session layers, and are called the application protocols. They define the actual physical components such as connectors and cable and how data is exchanged between systems. The type of network interface and the access method are also defined here.

The transport services, which are called the transport protocols, are only working in the transport layer. They provide connection-oriented data-delivery services across networks. Basically, transport protocols provide end-to-end data exchanges in which systems maintain a session or connection with each other for the reliable sequenced exchange of data.

The network services, which are called network protocols, are working in the last three layers. Those protocols provide link services for communicating systems. They handle things like addressing and routing information, error checking, and re-transmission requests. They also provide the procedures for accessing a network as specified by the particular network in use

such as Ethernet, token ring, and so on. Communication protocols are mainly affected by the transport services, and the network services layers, since this is the place where one application applies for transmission to another application located away from the first one.

Communication protocols have been developed for the purpose to gain many advantages for the transmitted data. The most known tangible advantages are: keeping some sort of standardization for the sender and receiver parts in the network; making sure the received piece of data departing save and correct; assuring data will be sent to the correct destination, and the most recent protocols insuring the piece of data selects the best and minimal path during its journey to the distention over the network [Black87].

B.4 Communication protocol layers

As been defined earlier, interconnectivity between systems has been defined in number of roles to govern the process of such data exchange. The main reason behind defining roles is to make the delivered product overall the world to be interoperable as much as possible. The Open System Interconnection (OSI) model is a standard defined by the International Organization for Standardization (ISO). It defines a layered architecture that categorizes and defines protocols for communication between end systems. During the communication session, each process running in each of the layers between the computers will communicate with one another. The following paragraphs give full details about the functions each of the seven layers handles, as well as, the points of communications among those seven layers.

The application layer acts as the window for the application process to access the OSI environment. It represents the services that directly support users and application tasks. This layer will put a definition in the packet define the type of operation that is required to be sent to the other site. There are many protocols defined by the OSI standard such as virtual terminal, file transfer, distributed transaction processing, message handling system, and directory services. So all the applications require networking features reside at this layer and access underlying communication protocols. The node address of the source user added to the data frame.

The presentation layer considered as part of the operating system, and the application the user run in the source workstation. In this layer information is formatted either for display or print. This layer relieves the application entities from being concerned with data representation to provide syntax independent while transferring data such as encoding data in a standard way, compress data to reduce the number of bits that have to be transmitted, and encrypt data for privacy and authentication.

The session layer coordinates the exchange of information in between systems by using conversational techniques. In that some applications may require a way of knowing where to restart transmission of data when connection is temporarily lost, or may require a periodic process to indicate the end of one data-set and the start of a new one. So this layer is the one responsible to enable two applications to talk to each other. It will establish sessions of the connectivity type between sites and synchronize between end user tasks by placing checkpoints in the data stream. When failure happens then there will be restart point, and finally the dialog control will take care that speaks, when, for how long, and so on.

The transport layer is the one responsible to ensure that data units are delivered error-free, in sequence, with no losses or duplications. This layer relives the higher layer protocols from any concern with the transportation of data between them. The type and complexity of the work done by this layer is dependent on the type of service it can get from the network layer next to it. This layer can provide connection oriented or connection-less services. In the first type, which is used for long transmissions, a circuit is established through which packet flow to the destination. In this arrangement, packets arrive in order and don't require a full address or other information because the circuit guarantees their delivery to the required destination. For the second type, the session does not establish circuits or provide reliable data delivery. Packets are fully addressed and sent out over the network. Each packet can conceivably follow a different path and even arrive out of order. The transport layer protocols at the

destination can put the packet in order and request a transmission of missing or defected packets.

The network layer is the one defines protocols for opening and maintaining a path on the network between systems. It is concern with data transmission and switching procedures, and hides all these things from upper layers. This layer also will determine routing method by looking to the packet addresses. If packets are sent to a workstation on the local network then it is forwarded directly without adding routing path to the packet. Otherwise, routing path will be added to the packet and will be sent to the routing device for transmission.

The data link layer sits just above the physical layer. Therefore it defines the protocols that directly interact with the physical components of the network such as the network adapters and cables. It controls the flow of information across the link, and adds its own error checking to the packets it sends out across the links. This layer is the point where external connections can be handled. Because this layer can provide error control, higher layers may not need to handle such services. However, when reliable media is used, there is a performance advantage by not handling error control in this layer, but in higher layers. All carrier services discussed previously operate at that layer.

The lowest layer, which is the physical layer, is the one responsible to take the packets prepared by the upper layers and sends them to the required destination. This layer defines protocols that describe the four aspects of the physical interfaces: (1) mechanical aspects are connector specifics, circuit-to-circuit assignments, and connector latches; (2) electrical aspects include voltage levels representing binary values and resistance; (3) functional aspects assign functions such as control, data, and ground to particular circuits; (4) procedural aspects define the procedures used to exchange data [Terplan87].

B.5 The communication protocol interoperability requirements:

Over the past two decades, numerous communication protocols have been developed such as SNA, OSI, TCP/IP, DECnet, AppleTalk, and many other protocols, where some are strong OSI compliance and some are not. Each of those communication protocols has its own advantages and disadvantages in terms of network performance, security, manageability, and application availability. On the other hand, the application-programming interface API used by each of the distributed systems is specifically bound to run on a single communication protocol. As the number of different installed communication protocols increases, this will increase the management complexity, as well as it will increase the complexity of performance tuning, bandwidth allocation, and other network considerations.

The model to be built is to provide a flexible vehicle for the communication and negotiation process that is necessary to support conflict resolution. The goal of the protocol is to progressively refine the context within which the reconciliation occurs, and to dynamically construct the reconciliation context, based on the knowledge that is available to the participating systems. The following figure explains the interoperability requirements by the Interoperable/Heterogeneous Communication Protocol Sub-Layer.

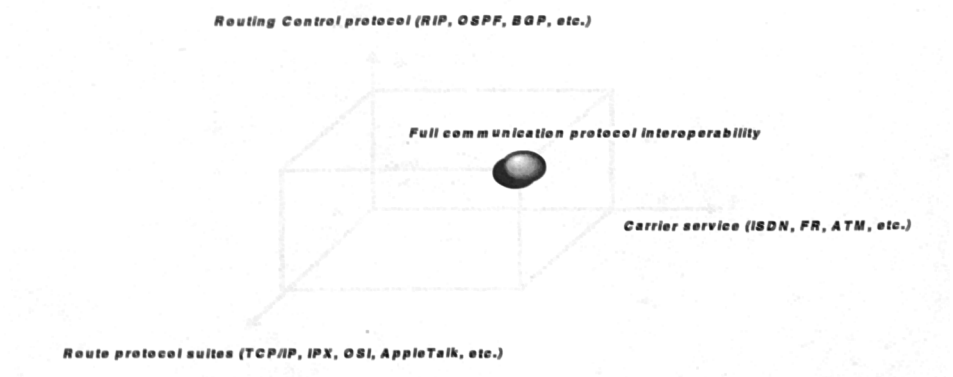


Figure B.1

As the above figure explain, whenever the communication protocol layer can combine between different route protocol suits as an interior protocols within the local network, different routing control protocols as exterior protocols in between autonomous systems, and different carrier services as for the interior and exterior packets delivery services, then this will lead to full communication protocol interoperability. Of course, full communication interoperability should keep the level of security into packets to its maximum as it is provided if a single communication protocol is used. Part of the layer knowledge base will be responsible of taking new protocols in the service and switch packets between different communication protocols. Communication protocols conversion will only fail if both sides of the connection do not get any knowledge about the other side communication protocols. In the case if both site IEs got the knowledge of switching communication protocol then the less critical site will handle the conversion and send the packet ready to the other site. If both sites got the same critical degree defined in its local IE then the IE in each of the sites will calculate the degree of busy time for each of the sites and will decide where the packet conversion will take place.

B.6 The design of the communication protocol layer

The basic responsibilities for communication protocols are the control of the physical devices within their domain and the control of the information flow within, through, and between devices. One of the most important goals in developing an enterprise system is to reduce the number of protocols in use, from say four or five to at most two or to develop a system that can learn and accept old and new protocols. Also, it is assumed impractical to have a single protocol to be used for wide and local area networks until such protocol is developed and used and be universally accepted. So, switching between different communication protocols will make interoperation between different systems an easy task.

In each site the IE will have a knowledge base capable of taking any packet and open it after doing all the necessary checking to make sure no errors in the transmitted packet, then covert the content of the packet in the destination database. The IE will support an increasing variety of applications, each with different requirements for security, integrity, bandwidth, response time, and dependability of services. IEs in all the sites will liaise with other sites to decide where to convert a packet if both sides' communication protocols are different from each other. Loosing no facilities provided by the communication protocol is one of the most important objectives of the layer. Now the packet conversion process is the core process of the communication protocol layer. This will be done by converting packets to accommodate working according to the roles assigned by different communication protocols with keeping the checking parameters, algorithms and any other factors of security to the level assigned by the original communication protocol.

Three possible scenarios can be defined for communication protocols in the wide area networks. The first is two networks using similar internal communication protocol with a different middle communication protocol. The second scenario is the same communication protocol used by both the local and wide area networks. The last scenario, which is the most complicated, is to have three different communication protocols used by the two local area networks and the wide area link. The following figure explains all the three possible scenarios.

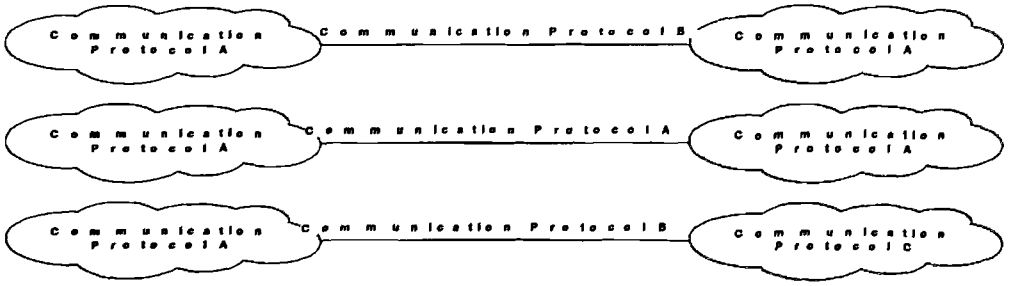


Figure B.2

In the previous figure, the first two scenarios have no impact on the design of the IE. However, the last scenario has to be treated so that interoperation between different protocols takes place. The communication protocol layer is the part of the IE, which will be responsible to switch between different communication protocols in a heterogeneous database environment. The following figure explains the overall process that will be handled by the communication protocol layer. If more details of the contents of the communication protocol layer knowledge base then an expansion to the three-dimensional boxes at the bottom of that figure should be included. At present these details are beyond the purpose of this part of the work. Such design will involve a very close study for as many communication protocols as possible to come up with common parameters for the knowledge base that will be responsible to resolve such conflicts.

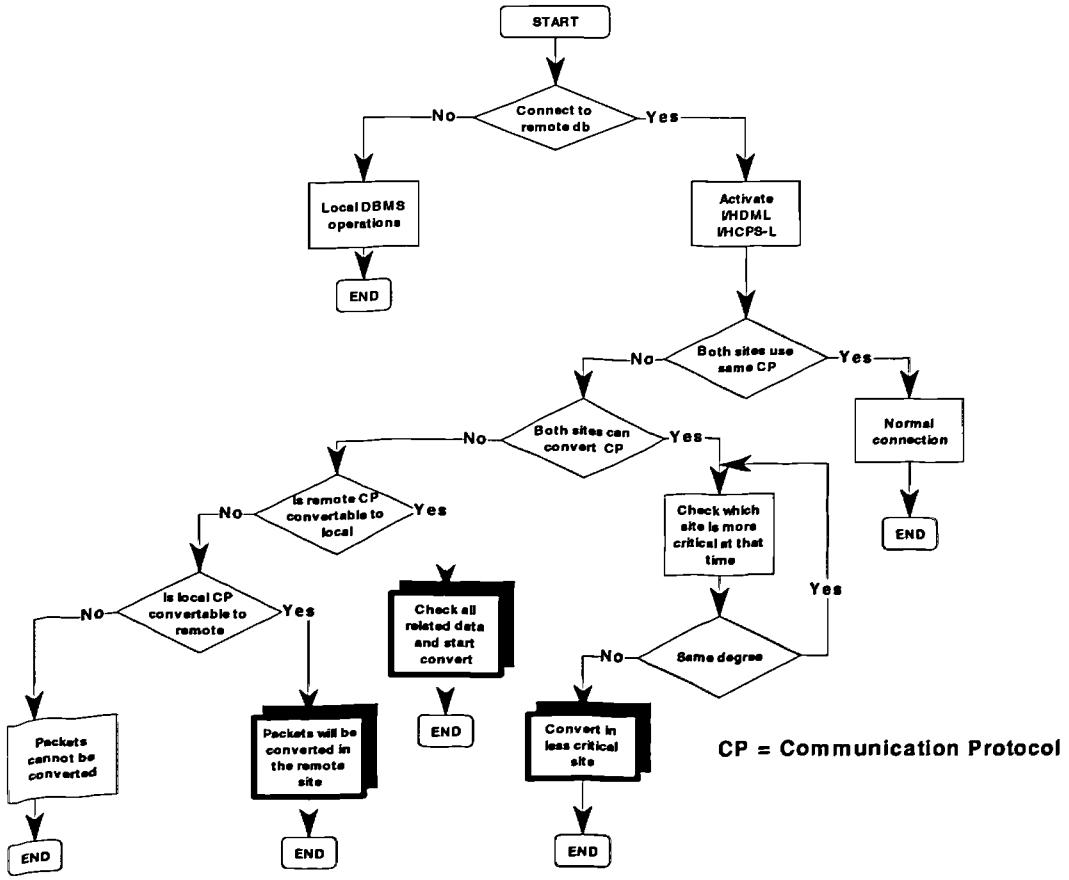


Figure B.3

One of the most important tasks the IE handling makes use of all the multi protocol converters either those are hardware devices or software based converters attached with the networks. IEs will be able to learn about other sites' communication protocols they are dealing with them. Such process will minimize the need for feeding the necessary parameters to the communication protocol layer knowledge base in a site. Also, if any of the sites feeds the required parameters to their communication protocol layers then this will let other IEs know about it. Such process will prevent duplication in efforts for converting packet to other protocols. Also this will prevent the knowledge base related to the communication protocol layer from being amplified.

The three-dimensional components defined in the previous figure B.1 needs to be looked after to include many parameters for the communication protocol's interoperation process. These parameters can be used as a definition for the flow control to prevent overflows and lost packets, acknowledgment of handled packets, sequencing of packets to insure correct

organization of them when arrive in the other place, check-summing mechanism to insure integrity of sent packets and retransmission process to ensure that corrupted packets will be sent again.

B.7 Background from the literature survey

As the number of the installed communications protocols increases, the complexities of managing the network and the operational costs also increase. Further, the interactions between the different protocols increase the complexity of performance tuning, bandwidth allocation, resource contention, and other network considerations.

The Multi protocol Transport Networking *MPTN* architecture is a general solution that breaks the binding between the application and the communications protocol, enabling users to reduce the number of installed protocols within a network. The *MPTN* gateway mainly solves two major problems. The first separates the application and the application support from the communications protocol. This separation will enable application running in different protocols to run over other protocols without any changes. The second is that *MPTN* concatenates networks running different protocols so that they act as a single network.

There are many multi protocol techniques that could be used as partial alternatives to the *MPTN* solutions. Some of these solutions are optimized to particular environments, some are simpler to roll out but more complex to manage, and others are more expensive. Some other solutions may be used to solve network connectivity problem that *MPTN* is not designed to solve, such as the communication between unlike applications. The following paragraphs explain the advantages and the disadvantages of the most will-known techniques used by the multi protocol techniques. Those are encapsulation, multi protocol routers, application gateways, and middleware.

Encapsulation is a general and widely used technique. It is mainly used when a packet is to cross from one network to another network using similar communication protocol and the middle network is different, as the situation in the first scenario of figure B.2. The primary advantage of such technique is simplicity. The application protocol can be wrapped in the transport protocol, and the transport protocol is unaware of the contents of the encapsulated message. At the destination the wrap can be taken off, and the application protocol that is left can be interpreted by the application, which is also unaware that a different protocol was used to transport the packet from one node to another. On the other hand, the disadvantages of encapsulation is that it may spawn unnecessary control traffic unless filtering is implemented and if such technique is built into the encapsulation, it must be done uniquely for each transport, which may involve some complex administration. This process will increase costs because intelligent filtering requires a deep understanding of the encapsulated traffic. Also, encapsulation requires complete execution of both protocol stacks which means that both networks must be managed separately and both networks must be configured completely.

Multi protocol routers are hardware boxes providing routing services for various protocols and additionally enable transmission of data for one or more communications protocols over a backbone network running a different protocol. This is mainly done through either an encapsulation technique or protocol translation for each pair of protocols in the router. Such technique is a convenient and efficient designed toward the routing function. Also, those can support existing application without changes and can be incorporated in a production environment without any disruption of ongoing operations. But because of the limited interoperability between different vendors this will give such technique a limitation. In such situation a router satisfies the to convert packet as in the second scenario of figure 2 and not the third scenario on the same figure.

Application gateways are application-specific solutions that connect two or more networks and translates data and messages between applications performing the same general function. Such application is the same as the one converting the email used by certain system to be readable on an email used by another system. Such technique got some disadvantages such as each application gateway must be tailored specifically for the pair of application. Also writing an application gateway from scratch requires full knowledge of both applications with

all error conditions to be taken into consideration. In addition such technique exist for a very limited applications overall the world.

Middleware solutions are those categories hiding the underlying network and its communication protocols from applications. Those include a specific product that can use various transport layer protocols. To use middleware, applications must be written to use the specific product as defined by the middleware vendor. So using middleware may involve changing the application to use the middleware, which is critical for applications that have been used without change for quite a long time [Pozefsky96].

MPTN tends to solve many of the problems discussed in the last multi protocol solutions that have been discussed. Specifically, *MPTN* tends to solve problem created by the third scenario presented at figure B.2.

B.8 Conclusions

The communication protocol layer is a structure for discussing multi protocol distributed computing solutions. The main functionality this layer will offer to the heterogeneous distributed databases' users is interoperation regardless of the communication protocol that is used by each of the interoperating sites. This alternative has strengths and weaknesses that can be magnified or reduced by any given situation. A typical network today will have a variety of protocols and often a variety of multi protocol solutions. The solution that the communication protocol layer proposes in this report is a purely software conversion between different protocols. Of the most important weaknesses that may be encountered by such converters is speed of conversion as well as the speed of encrypting and decrypting the packets. As it has been stated in a previous paragraph, a complete protocol converter is a huge task that will require a very deep study to the different communication protocols in use today. Such work by it self is a stand-alone research needs huge effort, which is above the scope of this research.

Appendix C

Analysis & Design Methodology for the IE

Abstract
Analysis is the study of a problem, prior to taking some action. It means the process of extracting the needs of a system, what the system must do to satisfy the client, and not how the system will be implemented. The study of analysis comes as the result of software expansions and complexity. The problem of software expansions and complexity makes researchers and practitioners revise the problem-definition phase to add structure to the entire process to manage and control software development. Many practitioners came up with ideas and methodologies in analysis, which have been defined as the process of breaking down the problems with the aim to simplify them for the coding step. The most popular are those given by Yourdon and DeMarco in the structured systems analysis and design methodologies, while Fusion, OMT/Rumbaugh, Booch, CRC, and Objectory are the most successful object-oriented approaches.

As it has been said, "Necessity is the mother of invention", analysis and design methodologies are aimed at helping to simplify the way people think about problems. In that object-oriented analysis and design methodologies complement the process of the previous analysis and design methodologies. They are consisting of mixture of the best steps applied in the previous analysis and design techniques. It is then said, objects better represent the world as people view it, rather than abstract the process to simplify the flow of data.

The aims of this part of the thesis is to discuss issues in the analysis and design methodologies, and recommend the approach that will be used for the analysis and design of the IE "Interoperable Engine" [Ashir2000]. Also the second aim is to discuss issues, which have already made benefits from object-oriented systems such as object-oriented languages and object-oriented operating systems.

C.1 Introduction

During the 1980's, hardware technology advanced considerably, which resulted in greater efficiencies and lower prices. By contrast, the software development process during the same period has been improved at a rate barely discernible. This situation has created a gap between the two technologies, leaving software designers unable to follow the speed and power of available hardware platforms. However, the development of object-oriented systems could place software development at the start of decreasing the gap in between the two technologies.

Systems analysis and design methodologies where introduced as the result of software expansion and complexity at the same time. During the analysis phase the problem is broken down into entities and relationship between these entities. The problem of software expansion and complexity makes it necessary for researchers and practitioners to revise the problem-definition phase to add structure to the entire process to manage and control software development.

The object-oriented analysis and design methodologies have evolved from the structured systems analysis and design. Yourdon, who is the inventor of structured systems analysis and design, describes object-oriented analysis and design as an extension for the structured analysis and design methodologies.

Object-Oriented software development, including object-oriented analysis, object-oriented design, and object-oriented programming, is a promising approach to developing software systems in order to reduce costs and increase flexibility in general during analysis and design phases by allowing reusability, extendibility, maintainability, and programming in large [Luker94]. Although alone it will not eliminate all the analysis problems (and hard work and dedication are still needed to produce the best and most efficient software possible) it is the most promising. Many software-engineering researchers have proposed an object-oriented approach to software design, because they see that this approach generally imitates reality better than traditional data flow or state transition design approaches. By this approach, which is the same in all the analysis and design approaches, the problem is broken down into entities and communications among these entities. The entities then conceptualized in a hierarchical manner to use the properties of inheritance. In many ways the analysis is the design in object-oriented analysis, once the problem is analyzed. Object-Oriented analysis technique involves modeling the process as seen by those who work with the system. This makes the object analysis process easier and straightforward than other techniques. Also there is another reason behind the success of object-oriented technique, which is the combination of the data description and operations performed on that data into one entity, which traditional analysis techniques provide separately. This enable objects to capture more information of greater importance about the process being modeled than virtually all other techniques [Coad91].

This part of the thesis proposes an analysis and design technique for the IE and in the light of the study discusses systems that benefit from the object-orientated techniques such as object-oriented operating systems, and object-oriented languages [Ashir2000].

C.2 Analysis and design methodologies

Analysis is the process of investigating how a particular business system currently operates, modeling the system and determining the essential characteristics of potential automated solutions. This involves often observing complex processes, interviewing people involved in the process and laying out the process as a data flow, state transition, or other pertinent modeling methods.

The main goal of the analysis phase is to build a problem model - that is, to create a description of just what exactly is needed. These may take the form of interviews, specifications as to level of performance [Yourdon93].

In this part a discussion of the structured analysis and design methodology and three of the object-oriented methodologies will be presented, with a mutual comparison.

C.2.1 Structured Analysis/Structured Design methodology

In the present time most of the analysis techniques in use are those based on data flow diagrams (DFDs). Here a discussion of the structured analysis/structured design (SA/SD), which is considered as one of the representations of the data flow approaches, will be given. Yourdon, Constantine, DeMarco, Page-Jones, and others have written about SA/SC. Ward and Mellor have added real-time extensions to the SA/SC. SA/SD is pervasive, applicable to many problems, and well documented.

SA/SD supports three orthogonal views of the system: object, dynamic, and functional models. It stresses functional decomposition. In that a system is viewed primarily as providing one or more functions to the end user.

SA/SD includes a variety of notations for formally specifying software. During the analysis phase, *data flow diagrams* (DFDs), *processes specifications*, a *data dictionary*, *state transition diagrams*, and *entity relationship diagrams* are used logically to describe the system.

DFDs are the focus of SA/SD methodology and they are model the information of data as it is flows through the system. DFDs where first used in the software-engineering field as a notation for studying systems design issues. In turn, the notation has been borrowed from earlier papers on graph theory, and it continues to be used as a convenient notation by

software engineers concerned with direct implementation of models of user requirements. DFDs components are *process specification, minispecs, data flow specification, data stores, and behavioral state transition diagrams and event flow specifications* [Rumbaugh91].

Process specification describes the type, number of instances and activation mechanisms for any type of process, and there is only one process specification for each process. Furthermore; process specification has got some components such as the name of the process, the meaning of it, the type of it (i.e. a control process, a data process, or a process group), persistence of the process and some others not necessarily fixed for the definition of all the processes.

Minispecs are provided to give a regress specification for each data process within the system. They are also used to specify control processes in certain circumstances. Minispecs must be precise, so that they can be easily converted to an understandable testable code, so that subject-matter specialist can understand them. The specification of the minispec should state the rolls that relate the outputs to the inputs. They also allow both external and internal specifications. External specifications are used to define the effect of the process, testing, and formal proof of correctness. On the other hand internal specifications are used to describe how the process will be built.

Data flow, which connects the output of a process to the input of another process, can be defined as a pipeline along which information of known composition is passed. Each data flow appearing in the DFD has its own specification. This type of specification is used to specify data flows and their components. Each data flow has a unique name that is used to specify it. This entry identifies the data flow being identified. No component of a data flow may have the same name as any other data flow or component of a data flow. Each data flow has a meaning, which gives the significance of the data item to the system. Every data flow has its own structure. The data flow structure shows whether the data flow value is an element, group, dialogue pair, or multiple. If the structure is group a composition is defined. The composition gives the contents of the data flow and the structure of one occurrence of the data flow. The composition of a data group is defined using the inclusion, selection, iteration, and optionally constructs. It is usually written out using a standard syntax for these constructs. This is described under each data composition structure. Data flow specification has much more components explaining the flow very deeply, preparing it for the final design and implementation, which is not part of this report.

Data stores are defined as the temporary holders of information for later use and they are not intended to be permanent. They cannot change the data they are holding, and what goes in is exactly what comes out. However, when a process accesses data contained in a database, it may access a subset, such as data elements that are part of a larger set of data elements. Data stores have some rules governing the entrance and retrieving of data such as: (1) one or more elements in a data store may be accessed; (2) data stores do not change any of the data left in their care; (3) data that enters a data store must eventually leave that data store; and (4) the accessing mechanism such as key is not depicted or documented. The data store specification defines which entities and relationships are included in the store. It is used to allow the crosschecking between a DFD and an ERD 'Entity Relationship Diagram'. The DFD shows stored information using a named store. This store contains information relating to one or more entities and/or relationships. The data store specification defines exactly which entities and relationships this store icon represents.

The behavioral State Transition Diagram (bSTD) highlights the modes of behavior of a system or portion of a system, what causes the system to change the modes of behavior, and actions that must be carried out to cause this change. The diagram also used to define the effects of events and conditions on the behavior of the system. bSTD has got some components such as action which carries operations such as begin, disable, enable, initials, signal, and trigger start. The other components are the name of bSTD, clock functions to determine the time, and other components such as comments, conditions, and connectors for the purpose of reducing graph complexity.

The last component of the DFD is the *event flow specification.* An event flow specification states the meaning of the flow and declares any other event flows that may be included in the flow. Event flows have some specifications like the name, which is a unique label for the event flow, the meaning that describes the purpose of the event flow, the structure, the persistence, and other event flows included if the structure type was multiple i.e. a grouping of event flows [Yourdon93].

The term *data dictionary* had developed a dual meaning. Among Database Management Systems vendors, the term refers to the catalogue of data items and their properties that the DBMS must manipulate during the run. This catalogue is oriented for the help of the DBMS, in that it holds the physical characteristics of data such as fields, data type, and the size of the fields, and not the meaning of them. On the other hand, data dictionaries of the analysis phase meet only the needs of this phase. Those needs include the definition of what data names mean, what other data items they are composed of, what the company policy is regarding that data, where each data item comes from, and where it goes to [Yourdon83].

On the other hand ER diagrams provide precisely what is needed to represent information relationships. They give an effective means of capturing the clusters or aggregates of information and the relationships these have to each other. The ER diagram, which is part of the DFD, shows the function of the system, showing the existence of one or more groups of stored data, but deliberately says very little about the details of the data. ER diagrams consist of two major components: object types and relationships. Object types are shown as a rectangular box and they represents a collection, or set of objects from the real world whose members play a role in the system that can be identified uniquely and can be described by one or more attributes. Relationships, shown as a diamond-shaped box on the diagram, are represented as a set of connections between the object types connected by arrows to the relationship [Peters88].

Subsequently, when the user implementation model has been completed, the job of system analysis is officially over. Everything exceeded this point talks about the design and programming and implementation and testing matters. The only work that has to be taken by the analyst tracks the design work to make sure that a full understanding of the designed system works its way to the target.

In the design phase, details are added to the analysis models and the data flow diagrams are converted into structure chart descriptions of programming language code. Structured design addresses low-level details, as example, data flow diagram processes are grouped into tasks and allocated to operating system processes and CPUs. Data flow diagram processes are converted into programming functions, and a structure chart created to show the procedure call tree [Rumbaugh91].

C.2.2 Objet Oriented Analysis & Design methodologies

Endless demand for better software development methodology that reduce both overall systems development and maintenance has led to the acceptance of object-oriented analysis and design methodologies recently. As an overall process, object-orientation tends to use the normal human thinking in expressing problems. Also, the methodology is considered as a new toolkit that can be added to the traditional approaches (such as dataflow, process flow, and state transition diagrams); and it is not in place to replace those traditional approaches. In recent days most client/server application development tools are emphases object-oriented features, because it is found to be very effective in business problems.

The boundary between object-oriented analysis and object-oriented design is not clearly defined in the literature. Some processes used by one author during analysis may be included in another author's design technique. Some authors said that object-oriented design could be considered as a superset of the object-oriented analysis phase. They define the object-oriented analysis as the one model the problem domain by identifying a set of semantic objects that interact and behave according to systems requirements. On the other hand, they define object-oriented design, which is suppose to be language independent to, as the part

models the solution domain which includes the semantic classes and other classes defining (interfaces, applications base utilities, etc.) identified during the design process.

Many course syllabi and textbooks subscribe to the notation that object-orientation requires nothing more than a change in language. But it is considered as a true paradigm shift in software engineering. It requires a complete change of worldview [Luker94].

The most important issue to figure in object-oriented analysis and design is the identification of classes, attributes, and methods or it is also called behavior. Relationships between classes are also part of the identification process.

There are three categories of object-oriented analysis and design methodologies that can be figured from the existing methodologies: 1) process only, 2) representation only and 3) process and representation. The first is the procedural methods supporting object-oriented analysis or design and not including any object-oriented analysis and design diagrams or notations. The second refers to graphical notations or diagrams for depicting the output of object-oriented analysis and design and focus on visually representing a design, and not on how to derive a particular design. The third is encompasses both processes for performing object-oriented analysis and design and representations for specifying the results. Next, at this comparison of three object-oriented analysis and design methodologies only category three will be considered [Monarchi92].

The next section is considered as a comparison of three analysis and design methodologies. Those are the Object-Oriented Analysis/Object-Oriented Design/Coad and Yourdon, Object-Oriented Modeling Technique/Rumbaugh, and the Fusion/Coleman. Those three analysis and design methodologies addresses more issues than other methodologies does. They are considered of the top used methodologies these days [Monarchi92].

C.2.2.1 Object-Oriented Analysis/Object-Oriented Design by Coad & Yourdon

Coad and Yourdon mention some key motivations in favor of OOA/OOD instead of using traditional analysis and design methodologies. These benefits can be summarized in the following points:

- Object-Orientation tackle wider domain of problems.
- It improves abilities of analysts during work in the problem domain.
- Internal consistency is increased from the start of analysis till programming phase.
- Clearly define commonality between classes and objects.
- Object-Orientation specification is flexible when change is required.
- Reusability of code is high and flexible in Object-Orientation.
- It provides strong foundation for analysis, design and programming phases.

Code and Yourdon mention that OOA/OOD, which is considered as an extension to the structured analysis and design methodology mentioned above, reduces the problem complexity and the system's responsibility within it. They stated also that no major difference between analysis and design phases, and no transition between them is required. They also stated that no waterfall model has to be followed, and that still different skills and strategies needed for analysts and designers. Their object-oriented approach consists of classes, objects, inheritance, and communication messages [Coad91].

The OOA consists of five major activities which are not necessary comes sequential. Those activities are finding classes and objects, identifying structures, identifying subjects, defining attributes, and defining services. According to those activities the OOA consists of five layers. On the other hand object-oriented design added four other components to the analysis part, and those are also not necessary comes in sequence. They are human interaction, problem domain, task management, and data management [Coad91].

In this approach reuse is one of the main issues, and it is behind why it is an important point to check previous OOA results in the same and similar problem domains. Here analysis and

design processes is achieved by following two activities where each consists of number of sub-activities and those are the object-oriented analysis and the object-oriented design [Coad91].

For achieving the first part, which is the analysis, it is important to achieve six sub-activities, and again regardless of the order. They are: Identifying class and objects, where the whole problem domain will be studied; Identifying structures, where kinds of structures such as (specialization, generalization, whole-part structures, and multiple structures) are identified; Identifying subjects, where achieved by promoting the uppermost class in each structure upwards to a subject, and searching for minimal interdependencies and minimal interactions between class and objects in different subjects; Identifying attributes, where many cases such as attributes with non-applicable value, class and objects with only one attribute, attribute with repeating values, many to many instance connection and many other cases equivalent to those are identified; Identifying services, where general services and messages connections are identified to the objects; and Prepare documentation, is considered as the last step in OOA, and it simply puts all the previous documentation together [Coad91].

The second part is the design part, and it consists of four sub-activities. Those are; Designing the problem domain component, and as the analysis part recommended at the start, this part is also recommend searching for previous design and classes that can be reused. Also this part is concerned with grouping classes, naming common set of services, some time changing problem domain to improve performance, isolating some low-level components in a separate class for simplicity purposes. The second designs the human interaction component, which is a kind of classification for the users according to the skill level, organizational level, and membership in different groups. After that existing human interaction systems are studied and detailed interaction is designed. To chick if the previous work meet the requirements then prototype is built, and finally after the detailed interaction is found to serve the needs, the human interaction class is designed; the third sub-activity designs the task management component [Coad91].

C.2.2.2 Object-Oriented Modeling & Design/Rumbaugh

This methodology state that the main difference between the traditional approach in software development and the object-oriented approach is the fact that the object-oriented approach is not based on functional decomposition but on describing the real objects that play a role in the real world. Systems can be best understood by studying their static structure, followed by focusing on the changeable i.e. dynamic changes of the structure. Aspects of a system concerned with time and changes are captured in dynamic models. Models are abstractions built to understand a problem before implementing a solution, which is the same as the flowchart in the SA/SD methodology, which is a preliminary explanation of the problem. All abstractions are subsets of reality selected for a particular purpose. The object modeling technique (OMT) uses three kinds of models to describe a system. Those are *object model, dynamic model, and functional model.*

An object model or static model, describes the objects in the system and their relationships. The object model is represented graphically with object diagrams containing object classes. Classes are arranged into hierarchies sharing common structure and behavior and are associated with other classes. Classes define the attribute values carried by each object instance and the operations, which each object performs or undergoes. Additional constructs for modeling associations include link attributes, role, qualifier, and aggregation. Also *propagation* or *triggering* is another technique offered by the object model, in that it is the automatic application of an operation to a network of objects when the operation is applied to some starting object [Rumbaugh91].

Object modeling offers inheritance as a powerful new technique. Inheritance has two different but complementary aspects. Those are extension, which means that a subclass may add new features to his super class, and restriction, which means that a subclass may, contains inherited features from the super class.

An added powerful tool for implementing complex systems called metadata, which is data that describe other data. Many real-world applications have metadata facility such as those in the

DBMS database table definition for storing information. Object classes are metadata, since they describe objects. Also, the candidate key, which is a term commonly, used within the database community can be defined within the associations and this can be a one or multiple key for each association. Furthermore a technique such as explicit constraints among objects, links, and attributes are some times needed to express application semantics. The notation for a constraint is a comment in braces near the constrained entity; a doted line can be added to bind constrained entities.

The dynamic model describes the interactions among objects and those aspects of a system concerned with time and the sequencing of operations in the system. The dynamic model is represented graphically with static diagrams, where each of them showing the state and event sequences permitted in a system for one class of objects. Dynamic modeling deals with flow of control, interactions and sequencing of operations in a system of concurrently active objects. The major dynamic modeling concepts are events, and states. Events are those things that happens at a point in a time and it has no duration such as flight departs from one place. An event is simply an occurrence that is fast compared to the granularity of the time scale of a given abstraction. Events are grouped in one class, by giving each one a name to indicate common structure and behavior although each one is a unique occurrence. An event conveys information from one object to another, and those conveyed data values are the event attributes. The state is an abstraction of the attribute values and links of an object. Sets of values are grouped together into a state according to properties that affect the gross behavior of the object. A state has duration; it occupies an interval of time and it is often associated with a continuous activity, such as the ringing of a telephone, or an activity that takes time to complete, such as flying from point to point. States and events can both be expanded into nested conditional state diagrams to show greater detail. They can both be organized as inherited hierarchy [Rumbaugh91].

The functional model is the last modeling tool and the one that describes the data transformations of the system. Specifically the functional model is the one describing computation within the system. The functional model specifies "what happens", which s the step after knowing "when it happens by", which is described by the dynamic model, and "what is happens to", which is described by the object model. Functional model show how output values in a computation are derived from input values, regardless of the order in which the values are computed. The functional model consists of multiple data flow diagrams. It is also includes constraints among values within an object model. The relation of the functional model to the other two is that, the functional model specifies the meaning of the operations n the object model and the actions in the dynamic model, as well as any constraints in the object model. The functional model is considered as the main model for non-interactive programs, such as compilers, which have a trivial, dynamic model, and their purpose is to compute a function. It consists of multiple data flow diagrams, which are graphs showing the flow of operations and the functional relationship of the values computed by the system, including input values, output values, and internal data stores. Data flow d'agrams are cons'sting of *processes* that transform data, *data flows* that move data, *actor objects* that produce and consume data, and *data store objects* that store data passively [Rumbaugh91].

Some times decis'ons may take place within the data flow diagrams such as Boolean, which is either true or false values are given in this case. Decision affects whether one or more functions can be performed rather than passing values to other functions. Those are part of the dynamic model, although they don't have input values from the decision funct'on. It is sometimes useful to include them in the functional model as a doted line from the process produc'ng the Boolean value to the process being controlled, as an example the password verification.

All models, object, dynamic, and functional, involve the same concepts, as data, sequenc ng, and operations, but each of them focuses on a particular aspect and leaves the other aspect non-interpreted. Al three of them are necessary for the full understanding of the problem, although the balance of importance among them varies according to the kind of appl'cation. The three models come together in the methods implementation, which involve data, control, and operations. The three models relate to each other by the following scenario,

"To the functional model: The object model shows the structure of the actors, data stores, and flows in the functional model. The dynamic model shows the sequence in which processes are performed.

To the object model: The functional model shows the operations on the classes and the arrangements of each operation. It therefore shows the supplier-client relationship among classes. The dynamic model shows the states of each object and the operations that are performed as it receives events and changed state.

To the dynamic model: The functional model shows the definitions of the leaf actions and activities that are undefined with the dynamic model. The object model shows what changes state and undergoes operations [Rumbaugh91]."

Last part discussed the OMT concepts, specifically the concepts and notation for the object, dynamic, and functional models. Here a discussion of the process for devising the object modeling technique will be presented. The object modeling technique consists of three phases: the analysis, the system design, and the object design.

The analysis is concerned with understanding and modeling the application and the domain within which operates. It will use the real world model consists of the object, dynamic, and functional models. The analysis model service several purposes: It clarifies the requirements, it provides a basis for agreement between the software requester and the software developer, and it becomes the framework for the later design and implementation. The analysis process can be defined as the problem statement preliminary description. This description is generated from the requests of users, developers, and managers and those are known as requesters. By using the problem statement description, the preliminary model building is done with the support of users interviews, domain knowledge, and supposes the real word experience. The building of the preliminary models will be consisting of object models, dynamic models, and functional models. All the three sub models are not equally important in every problem. Almost all problems have useful object models derived from real word entities. Problems concerning interactions and timing, such as user interfaces and process control have important dynamic models. Problems containing significant computation, such as compilers and engineering calculations have important functional models. Analysis is not a mechanical process, in that the analyst must contribute with the requesters, or previous analysis knowledge. Analysis has not had a precise sequence, in that it is almost start defining the large problems as a bottom-up process. Now after obtaining an initial description of the problem, the analysis process can be summarized in the following four steps:

1. Building an object model, which identifies object classes, defining a data dictionary describing classes, attributes, and associations, adding associations between classes, adding attributes for objects and links, organize classes by using the inheritance, and grouping classes into models. Object model can be defined as the object model diagram plus the data dictionary.
2. Develop a dynamic model from scenario of typical interaction sequences preparation, identifying events between objects, prepare an event flow diagram for the system, develop a state diagram for each class that has important dynamic behavior, and checking for consistency and completeness of events shared among the state diagram. So the dynamic model is the state diagram plus the global event flow diagram.
3. Constructing a functional model from the input and output values identifications using data flow diagrams as needed to show functional dependencies, describing what each function doing, identifying constraints, and specifying optimization criteria. The functional model is the data flow diagrams plus the constraints.
4. This step is not more than verifying, iterating, and refining the pervious three models. Verification is the checking of classes, associations, attributes, and operations consistency and

completeness. The analysis document can be defined as problem statement, plus the object model, plus the dynamic model, plus the functional model.

The system design is driven by relevance to the computer implementation, so that it is understandable as to encode it to the machine. This is in contrast to the analysis phase which is suppose to be a meaningful information from the real world and to be understood by the requester. The overall architecture of the system is determined at this phase, and by using the object model as a guide the system is organized into subsystems. It is the phase of how the system will be done, and it will start from high level, than it will increasingly go to detailed levels. Breaking down the system into subsystem give the chance to a group of developers to work down the system, which means better system than a one designer work. In brief, at this phase the system designer must make the following eight decisions:

1. Breaking out the system into organized subsystems, in that each subsystem encompasses aspects of the system that share some common property such as similar functionality, the same physical location, or executing on the same kind of hardware.
2. Concurrency identification can be done in a single processor. For example, two subsystems are inherently concurrent; they need not defined as separated hardware, in that they can both be processed at one time by a single processor. This is the purpose hardware interrupts, operating systems, and tasking mechanism is to simulate logical concurrency in a uniprocessor [Bacon93, Jung97].
3. Deciding how the hardware will handle the software subsystems does allocating subsystems to processors and tasks. Decisions about the number of hardware processors are done at this stage, and the necessity of fast reaction from the processor side is decided as well at this stage.
4. Choosing an approach for the management of the data stores, in that data store for subsystem is chosen to give some sort of permanence for the application data.
5. Handle access to global physical and logical resources, and determine a mechanism to access them. Global resources include; processors, table drivers, disk spaces, etc.
6. Choosing software control implementation; in that software system has two kinds of software controls: external control, and internal control. External control is the flow of externally visible events among the objects in the system. Objects generate internal control operations as part of the implementation algorithm. Their response patterns are predictable. Most internal operations can therefore be thought of as procedure calls, in which the caller issues a request and waits for the response. Other software control implementations are available as well, such as role-based systems, logic programming systems, and other forms of non-procedural programs. Hear the control style is replaced by declarative specification with implicit evaluation rules, possibly non-deterministic or highly convoluted.
7. Handling boundary conditions, which must be considered by the system designer such as initialization, termination, and failure.
8. Setting trade-off priorities is an important matter. The system designer must set priorities according to the trade-off during the rest of design. He is often required to choose among desirable but incompatible goals. The designer at this stage must decide which goals are of highest priority and work them out.

Several kinds of systems are frequently encountered for which standard architectural frameworks exist. These includes two kinds of functional

transformations: batch computation and continuous transformation; three kinds of time-dependent systems: interactive interface, dynamic simulation, and real-time; and a database system. Each framework has got its own steps, or proposed steps to be followed. Most application systems are usually a hybrid of several forms, possibly one for each major subsystem. Other kinds of architecture are possible. System design can be described as the structure of basic architecture for the system as well as high-level strategy decisions.

The object design is a detailed description of the analysis phase. It is considered as a shift from application domain concepts towered computer concepts. It is mainly depending on the objects discovered during the analysis phase. The object designer at this phase has to choose among different ways to implement them, so that they act at their best situation from the execution time, memory, and costs point of views. Object oriented design is a preliminary process of refinement or adding detail. In brief during object design the designer must perform the following eight steps:

1. Combining the three models to obtain operations on classes and this is considered as the start of the physical organization of the program.
2. Design algorithms to implement operations to minimize cost of the whole system.
3. Optimize access paths to data, where the designer is suppose to strike an appropriate balance between efficiency and clarity.
4. Implementation of control, where in that state-event interactions that has been presented in the dynamic model, can be implemented using one of three different styles of control: Use of the location within the program to preserve the control state, explicit state machine representation, or concurrent tasks.
5. Adjustment of inheritance, where the definition of classes can often be adjusted to increase the amount of inheritance.
6. Design the implementation of associations, where the designer will analyze the traversal of associations, and implement each association as a distinct object or by adding object-valued attributes to one or both classes in the association.
7. The exact representation of the objects must be chosen. At some point, user defined objects must be implemented in terms of primitive objects or data types supplied by the programming language. Some classes can be combined.
8. Physical packaging, where the programs are made of discrete physical units that can be edited, compiled, imported, or otherwise manipulated. Information hiding is a primary goal of packaging to insure that future changes effects few modules. Modules should be coherent and organized about a common theme.

All those decisions in the final should be properly documented for simplicity reasons. Simply object design is a detailed object model, plus detailed dynamic model, plus detailed functional model.

Recently OMT/Rambaugh and Booch are joining their forces at Rational Software Corp., with the goal of unifying their approaches into an industry-standard, open methodology [Frank95]. The following is the process of steps in Booch:

- Identify the objects and classes that form part of the system.
- Construct the interface for classes, which identifies the semantics.
- Find relationships between the classes. This may cause new interfaces to be discovered, so apply the previous step again.

- The implementation stage, and the decision of representing attributes and behavior.

The following figure C.1 shows the cycle of the whole process in Booch methodology,

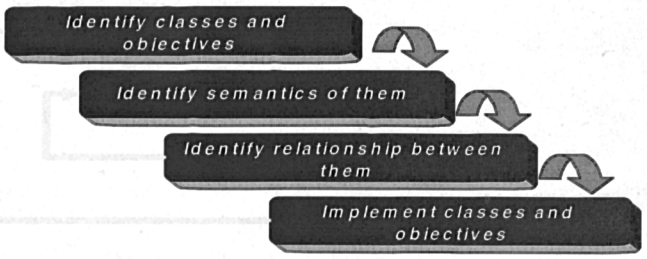


Figure C.1

C.2.2.3 Fusion/Coleman

Fusion analysis and design methodology in the other hand supports both the technical and managerial aspects of software development. It provides a divided and systematic process, which is divided into phases and sub-phases as the one of the OMT. Fusion is not restricted for new system development, but it can be used for re-engineering process of existing systems. It is also following the three steps, which are followed by SA/SD, and those are the analysis, design, and implementation. Fusion does not have a requirement phase. Recall that requirements have to be submitted by the requester, who will supply the initial requirement documents.

Analysis phase in Fusion produces two models capturing the aspects of a system. The first is the *object model*, which defines the static structure of the information in the system. The second is the *interface model*, and this defines the input and the output communication of the system.

The object model is based on extended entity relationship notation and it describes the structure of a system, but not its behavior. It can represent classes of objects, attributes and relationship between them by using tools almost the same as those used by the OMT/Rumbaugh. Also the extensions of them permit the use of aggregation and generalization, which are also equivalent to OMT/Rumbaugh [Coleman94].

The interface model describes the behavior of the system. It describes the sequence of the process for a system, and showing the flow of data among this system. In Fusion the interface model uses two models to capture different aspects of behavior. The first is the *operation model* that characterizes the effect of each system operation in terms of the state change it causes and the output events it sends, and this is analogous to the functional model of the OMT. The second is the *life-cycle model* that describes the behavior from the wider perspective of how the system communicates with its environment from creation until its death, and this has been adapted from Jackson System Design. The behavior of a system is described as a table of life-cycle model and operation model taken together [Coleman94].

Analysis in Fusion is an interactive and incremental activity. Throughout the phase, it is necessary to construct and use a data dictionary. The main data dictionary columns are, the name of the entity, the kind i.e. class, system operation, agent, etc., and the text which defines or explains the entry.

The process of analysis in Fusion can be done throughout four steps. The first step is to develop an object model for the problem domain. This object model will give precise definition for the classes and the relationship between them within the system.

The second step determines the system interface, which is consists of defining agents, system operations, and events. The agents are those active entities cooperate with an upper active entity. The system interface determines how functionality is mapped to individual

operations. The purpose in which this is done enforces a protocol in any agent that uses or communicates with the system. Consequently system interfaces have to be designed rather than discovered. Designing the interface requires technical decisions concerning the amount of communication traffic and response times. For those parts of the interface that communicates with human users, their needs must be considered as well.

A model called the system object model is a refinement of the object model, is concerned with showing the system boundary. In that a class or relationship that lies outside the boundary is not needed to carry out the functionality of the system.

The third step in the analysis phase is the development of the interface model. This model comprises a lifecycle model and an operation model. The order is not necessary, while it is only recommended to start with developing the life-cycle model first, because the lifecycle can be an aid to developing the operation model schemata.

The aim of the lifecycle is to generalize the scenarios. Life-cycle expressions can express repetition, alternation, optionally, as well as concatenation. The process for forming the life-cycle model will come as generalizing the scenarios to form named life-cycle expressions, then combine the life-cycle expressions to form the life-cycle model.

On the other hand the operation model defines the semantics of each system operation in the system interface. The preconditions and post-conditions for each process will be defined in the system. In that as long as the precondition of the process is true, then the post condition or the output has to response as the stated results. This kind of job has to be done by the analyst to satisfy the needs of the customer.

Each process within the system is defined as a separate schema. The schema contains the operation name and the description of this operation as the heading. Another five elements for the schema has to be defined as, the *Reads*; which will contain the input elements, the *Changes*; is the changed items within the schema, the *Sends*; is the sent data, the *Assumes* and *Results*; are preconditions and post-conditions consequently.

The process of developing a schema can be summarized as the following:

1. Develop the *Assumes* and *Results* clauses.
   - Describe each aspect of the result as a separate sub-clause of *Results.*
   - Use the life-cycle model to find the events that have to be output in R*esults.*
   - Check that *Result* does not allow unwanted values.
   - Add relevant system object model invariant to *Assumes* and *Results.*
   - Ensure *Assumes* and *Results* are satisfactory.
   - Update data dictionary entries for system operations and events.
2. Extract *Sends, Reads,* and changes clauses from the *Results* and *Assumes.*

The final step in the development phase checks the analysis models. It is unpredictable to know the accuracy of the analysis phase and how errors free it is. The designer will probably not worry if few trivial errors occur. However it will be waist of time and money in the next stages.

Two aspects of checking are recommended for the analysis models. They should be *complete* and *consistent.* The model is complete when it captures all the meaningful abstractions in the domain. So, completeness at this stage means the one against

requirements, and that no outstanding issues still with the customer. This kind of checking will make sure that all possible scenarios are covered by the life cycle, a schema defines all system operations, all static information is captured by the system object model plus any other related information. The model is consistent when there is no overlapping in between the analysis models. This will check that all classes, relationships, and attributes mentioned in the operation model appears in the system object model, and that all other concepts such as predicates are defined in the dictionary. The other check element is to make sure that the boundaries of the system object model is consistent with the system interface given by the life cycle model. Also all the system operations in the life cycle model have a schema, and finally all identifiers in all models have entries in the data dictionary. The following figure C.2 shows the whole checking process.

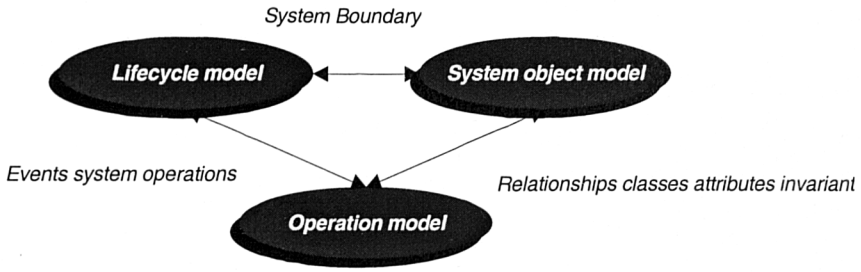


Figure C.2

At the *design stage* in Fusion, the purpose of it is to define how functionality the system is implemented from the previously defined *system object model, which* describes the classes in the system and the semantic relationships that must hold between them. The output of the design is a collection of interactive objects that realize the final operation model of the analysis phase. During this phase of the method four models are developed.

The first is the *object interaction graphs,* which defines the sequences of messages that occur between collections of objects to realize a particular operation. The system operations at this stage are taken from the operation model, which get the behavior of the operations and build up the object messaging structures that realize the abstract definition of behavior. The object interaction graphs are represented as boxes linked by arrows. The boxes are the design objects, and the arrows are the message passing. The main box is the one that got an arrow that does not come from any other box. The other boxes are called the collaborators. Traversing at this stage is a straightforward process.

The main components of the object interaction graphs are the objects. They can be defined as a single object, which is a solid box has the object name and the class name, or they can be defined as a collection of objects in a class, which is a dashed box with same content of the previous type. The objects in a collection may change over time.

The connection between objects identified as directed arrows from sender to receiver. Those are realized as a function or method call. Messages can be passed to single object class or a collection of objects in a class, or visa versa. Numbers to insure full control can sequence messages between objects. A definition to a new object can be defined by entering the word new before the object name. Next develop the object interaction graphs. Graphs are built for each system operation. They specify how the required system functionality identified during analysis and implemented by the objects in the system. Four steps are involved in the definition of the object interaction graphs. First, identifies the definition and implementation objects with each other. Second, implement the role for each object in computation by identifying controller and collaborators, which collaborate with the controller to implement the system operation. Then decide on the messages between objects. And finally, record how the identified objects interact on an object interaction graphs. Refinement of the object graphs is done in the final to check that each of the classes in the system object model is represented

in at least one object interaction graph. Also, the functional effect of each object interaction graph satisfies the specification of its system operation given in the operation model.

Second step in the design stage defines a *visibility graph* for each of the classes. These show how the object-oriented system is structured to enable object communication. They are constructed during three steps. The first inspects all the object interaction graphs. Each message on an object interaction graph implies that a visibility reference is needed from the client class to the object server. Second is to decide on the kind of visibility reference required taking into account lifetime of reference, visibility of target object, lifetime of target object, and mutability of it. Finally draw a visibility graph for each design object class. Then for each relation in the system object model check that there is a path of visibility for the corresponding classes on the visibility graphs, and that exclusive target objects are not referenced by more than one class and that shared targets are referenced by more than one class.

Next step the design phase is class descriptions. At this stage an extraction to the system object model, object interaction graphs, and visibility graphs is made to build class descriptions for the system operations. Each class description will record methods and parameters from the object interaction graph, data attributes from both the system interaction model and the data dictionary, object attributes from the visibility graph for the class, and inheritance information from the inheritance graph respectively. Checking at this stage is the process of making sure that all methods from object interaction graph, all data attributes from the system object model, all visibility references, and all inherited super-classes are recorded at this pint.

The fourth stage in the design process is to build the inheritance structures by looking at the classes to identify commonalties and abstractions. As in OMT this process identifies super-class and its subclasses. The process at this stage will look for generalizations and specialization from the object model, common methods from object interaction graphs and class descriptions, and common visibility from the visibility graphs.

The final step at this stage will require updates to the class descriptions with the new inheritance information to the previously defined system object model, object interaction graphs, visibility graphs, and class description. Finally, the implementation process is done in three stages: (1) coding, (2) performance monitoring, and (3) review to the coded system as subparts to monitor the overall performance.

C.2.3 Comparison of object-oriented methodologies

Object-orientation facilitates systems development by allowing systems developers to reuse past solutions. Through inheritance, objects can be reapplied in new development efforts. It also decreases the cost of applications maintenance. Its inheritance and encapsulation features help systems evolve gracefully. Because programmers can comfortably remove obsolete solutions of code and data designs, and maintenance is almost as easy as creating new object.

Organizations that can master object-orientation techniques can gain the productivity benefits that come with extensive systems modularity and program reusability. However, these organizations must have a development approach that uses the strengths of object-orientation.

During the past years object-orientation practitioners mention several advantages for this technology. Some of the most important are greater systems productivity in general or programming in the large, and higher reusability which increases productivity and quality of systems, maintainability and simplicity of it, and give an opportunity for better system design.

The reusability process is measured by the quality and quantity of the usability of the pre-tested pre-developed software components. This is of course will effect productivity and overall system quality. The use of existing, pre-developed, pre-tested, and documented objects requires less effort, low level testing and less custom documentation. Efficiency is improved because much of the expensive and error-prone development work is eliminated. Instead reuse promotes rapid prototyping, which is considered important in efficient software

development. Reuse is promoted by object orientation in two basic ways. First, the inherited features allow new classes of objects to use structures and methods developed for other classes. Second, incorporation of object libraries and catalogues for new systems can be either directly or with incremental modification [Weinberg92].

The maintainability in object-orientation is measured by the easy in step-wise bases, and the time taken for maintaining any part of a system built using the object-orientation technology. Also object-oriented systems are easier to maintain than its traditional counterparts because they can facilitate system maintenance through their inherent modularity and structure, and through from their natural insulation of object from each other. The modularity and structure inherent in object-orientation provide greater data integrity. This is also because each object operates on its own internal data through its own methods. Because objects are insulated from each other, maintenance of software pieces is greatly simplified, and changing in any part of the system such as the data structure, introduction of new data types, etc. is transparent to their user [Weinberg92].

As an overall, the object-oriented approach provides increased modeling flexibility and expressive power enabling of complex software development. It is also emphasizing the design phase of the systems development, helping to reveal problems early in the design process. In general object-oriented analysis and design has got several facilities over the traditional analysis and design methodologies. It integrate the analysis and design phases based on entity and class relationships, stabilize the development process, provide the specific procedure to identify attributes and operations, provide the formal textual document, and clearly describe the behavior of a system and classes [Chen94].

Both the object-orientated methodologies and traditional methodologies have much in common and use similar modeling constructs and support the three orthogonal views of a system which are, the information, time, and function diameters. The difference between them is primarily a matter of style and emphasis. In the structured analysis/structured design approach the most important dimension comes the functional, then comes the dynamic, and finally comes the object model. In contrast, the object-orientation use the object model as the most important, then comes the dynamic, and then comes the functional. The following table compares the three previously discussed object-oriented analysis and design approaches:

	Booch	Coad and Yourdon	Rumbaugh
1. OOA PROCESS			
Semantic classes	*	*	*
Attributes	*	*	*
Behavior	*	*	*
Generalization	*	*	*
Aggregation	*	*	*
Other	*		*
Placement of Classes		*	
Placement of Attributes		*	
Placement of Behavior	*		*
Dynamic behavior	*		*
2. OOD PROCESS			
Interface classes		*	
Application classes			
Base/utility classes			*
Optimization of classes	*	*	*
3. REPRESENTATIONS			

Table 1

Static Objects	*	*	*
Static Attributes	*	*	*
Static Behavior	*	*	*
Generalization	*	*	*
Aggregation	*	*	*
Other relationships	*		*
Dynamic communication	*	*	
Dynamic Control/Timing	*	*	*
Constraints on structure	*	*	*
Constraints on dynamic behavior	*		*
4. COMPLEXITY MANAGEMENT			
Structural	*	*	*
Behavioral			
Representation of static structure	*	*	
Representation of dynamic behavior			

Table 1 Cont.

C.3 Object oriented operating systems

The current trend of work towards building a full object oriented operating system. This will contribute very strongly in the software building crises. In that it will complete the circle of object orientation. Full compatibility between object-oriented software, as well as fast and easily modifiable software ICs will be created at this time.

C.4 Object-Oriented Programming Languages

Object oriented programming technique can be defined as the unit of modularity, so those workers produce sub components instead of complete solutions. The sub components are controlled by standards and can be interchanged across different products. By this technique programmers no longer build entire programs from raw materials. Instead, they produce reusable software components by assembling components of other programmers. These components are called software ICs to emphasize their similarity with the integrated silicon chip. Object oriented programming is ideal for a component-based architecture, because the encapsulation technique exposes only the parts really the one need to know about. And also it gives the flexibility to change the parts don't know about without affecting the body of the code. Object oriented programming is more than learning a new syntax. It means a new approach to design, as well as implementation. The object oriented programming offers some considerations for the implementation of an object-oriented design, those are: classes definitions, creating objects, calling operations, using inheritance, and implementing associations. On the other hand, object oriented design can still be implemented in conventional programming languages but required programming discipline, such as classes can be implemented as records in any modern language. The implementation of object oriented programming into non-object oriented languages face some shortcoming and problems. First, price is high for mapping from object orientation to traditional languages. Second, manually traversing class hierarchy when calling methods or passing arguments. Third, manually ensure error free by reinitializing new objects with their classes. And fourth, if changes are made to the object declarations then the programmer must determine their effects on the code and modify it accordingly [Budd91, Cox87].

Object-Oriented programming languages simplify the problem of maintenance. They provide a way of integrating data definitions and processing rules into cohesive modules called objects. Each object in an object-oriented programming system is a freestanding entity that can be combined with other objects to form a complete application [Kerr92].

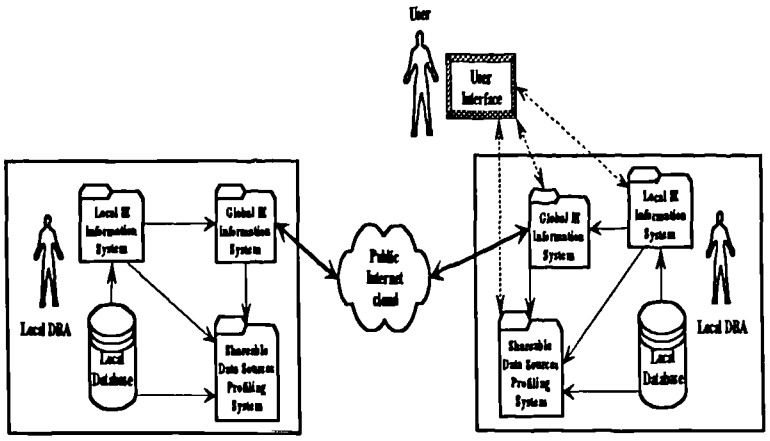


Figure 4

3.2 Exchanging information between participating IEs

The local IE information system contains a definition of a complete local systems. A further definition for the shareable parts of the local systems are defined as routes and only these parts are transmitted to the global IE information system. In each of the cooperating sites the global IE information system link the local shareable subsystems with the other global cooperating databases. Any alteration in the global shareable space is mainly done locally in the local IE information system and then migrated to the global IE information system. This stipulation is for the purpose to make the global IE information system maximum reliable and to make sure no long intervention is to occurs in the cooperation responsible subsystems. The initial partial local IE information system UML based classes are shown at appendix A figure A.1 [3, 13]. Accordingly, figure A.2 shows the global IE information system.

The shareable data sources profiling system is where database profiles are created and accessed by the local application programs. Basically it links information about data sources from both the local and global IE information systems.

Although, the design of the IE in its preliminary stages, this will involve assigning an information handling strategy between the cooperating data sources by which the exchange of the updates for the necessary metadata between the information sources can be done. The strategy should guaranty the maximum reliability and minimum intervention to the interoperation services.

4. Description of Participating Components in the IE

The actual building infrastructure of the IE is assumed to be the Internet by which nobody owns the backbone. Figure 3 has shown the overall architecture of the different IE components and how they are liked together. As shown in the figure, the main components are the Interoperation Engine layer, the heterogeneous databases repositories and the metadata proxy server.

Subsequently the *IE* local knowledge consist of three interrelated components of which each plays an important rule in the success of the interoperation mission of the distributed heterogeneous databases. The three interrelated components are (1) *users and shared systems profiling*, (2) *heterogeneous schema management*, and (3) *cooperating schemas profiling*. Those are explained graphically at appendix A figure A.1.

4.1 User and shared systems profiling

The IE system supports two type of users. These are the ordinary information consumer and the IE administrator. The one who will mostly benefit from the IE is the ordinary information consumer. The administrator is the person who will create users profiles, systems profiles and access profiles. User profiles are the grouping of the local and global users according to local site policies. Systems profiles are the same as user profiles but on the local and global systems. Access profiles are the link between a user profile and systems profiles. The following example explains the profiling technique in the IE system including user management. Figure 5 shows the relationship between the class diagrams for both the local and global profiling management subsystems. Details of the participating components shown at appendix A figures.

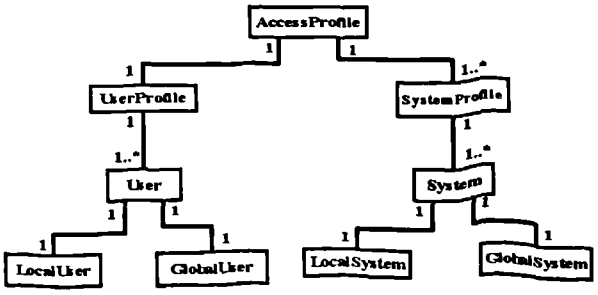


Figure 5

C.5 Conclusion

There are many reasons why data flow approaches are in such wide use today. Some of them are because programmers are still thinking in functions, so this makes functional or data flow approaches are easier to learn. Also because SA/SD is historically the first well thought out to software and system development, it is still looks easier to be used in thinking about problems. But it is well believed that the Object-Oriented thinking approach is the one that will apply the promises of easy and flexibility. This approach is the reason behind adding the most well known facilities to the programming languages, which are abstraction and inheritance.

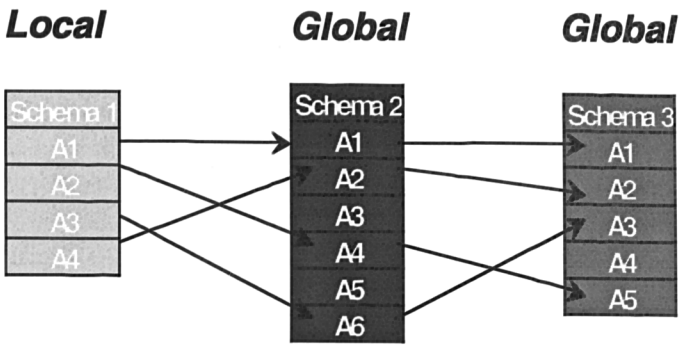
When comparing OMT with other object oriented analysis and design technique, it will show that the OMT is compatible with the few publications encountered today, and absolutely does not contradict with them. This technique is considered to be a consolidation of past efforts with some incremental improvements. The primary strength of OMT is its analysis phase. It has a well-defined process, and all the models use concise and understandable notations.

The OMT is similar to the object oriented analysis presented by Coad and Yourdon in the year 1990. Also, it discusses and gives some extensions to the methodologies given by Grady Booch "Object Oriented Development/Object Oriented Design" of 1986 and 1981, Bertrand Meyer "Object Oriented Software Construction" of 1988, Sally Shlaer "The Object Oriented Method for Analysis" of 1988, and Ivar Jacobsen "Object Oriented Development in an Industrial Environment" of December 1987.

On the other hand, as it has been discussed earlier, the Fusion method integrates the best aspects of several methods. It integrates the object model and process of the OMT, the object interaction of the CRC, the pre- and post-conditions of FORMAL METHODS, and the visibility of BOOCH [Coleman94]. This concludes that a combination of object-oriented analysis and design methods can be used for the analysis and design of the IE.

Appendix D

Information sources cooperation example



Schema

Code	Name	SubSystem Code	...
sc1	Schema1	ss1	
sc2	Schema2	ss2	
sc3	Schema3	ss3	

Sub System

Code	Name	Path	...
ss1	system 1	//aaa/.../../.	
ss2	system 2	//bbb/../../	
ss3	system 3	//ccc/../../../.	

Attribute

Code	Name	Schema Code	Type	Size	...
at1	.....	sc1	.....	.....	
at2	.....	sc1	.....	.....	
at3	.....	sc1	.....	.....	
at4	.....	sc1	.....	.....	
at1	.....	sc2	.....	.....	
at2	.....	sc2	.....	.....	
at3	.....	sc2	.....	.....	
at4	.....	sc2	.....	.....	
at5	.....	sc2	.....	.....	
at6	.....	sc2	.....	.....	
at1	.....	sc3	.....	.....	
at2	.....	sc3	.....	.....	
at3	.....	sc3	.....	.....	
at4	.....	sc3	.....	.....	
at5	.....	sc3	.....	.....	

System Starting Pointers

Profile Name	Starting Pointer
ssp1	scp1

Schema Cooperation Profile

Reference	Source Schema Code	Operation	Target Schema Code	Next Reference	...
scp1	sc1	Unification	sc2	scp4	
scp2	sc4	Intersection	sc7	ss20	
scp3	sc5	Difference	sc9	0	
scp4	sc1	Unification	sc3	0	
scp5					

Attribute Cooperation Profile

Code	Source Schema Code	Source Attribute Code	Target Schema Code	Target Attribute Code	...
acp1	sc1	at1	sc2	at1	
acp2	sc1	at2	sc2	at4	
acp3	sc1	at3	sc2	at6	
acp4	sc1	at4	sc2	at2	
acp5	sc1	at1	sc3	at1	
acp6	sc1	at2	sc3	at5	
acp7	sc1	at3	sc3	at3	
acp8	sc1	at4	sc3	at2	

Appendix E

Published Papers

Technical Perspective on the Heterogeneous Databases Interoperability

Jehad Saleh Ashir
Computer Services Directorate
The Central Statistics Organization
State of Bahrain, P.O. Box. 5835
Jehadashir@bahrain.gov.bh

Abstract

The rapid growth in the number of users of global networks such as the Internet, the number of systems used by those users, the amount of the available information, and the growth in the bandwidth has made new demands towards finding new techniques for interoperating between the enormous information pools that already exist. Many people from companies, government, schools and individuals will have access to an increasing amount of data. Indeed, users have already started to access huge amounts of information from disparate sites. Thus, the problem typically associated with the (1) dispersal nature and size of the available data, (2) the heterogeneous nature of the data resources.

The first problem, which is the dispersal nature and available size of information sources, is caused by many factors. The main one is the rapid connectivity of the local area networks, which forms the global backbone of the public Internet. Additionally, the success of linking the three-tier architecture (mainframe, mini, and Local Area Networks) has enlarged the size of the problem. Instead, this has increased the number of databases becoming available over the Internet. Also, the advances in linking heterogeneous communication protocols maximize the size of the available information source.

The second problem, which is the heterogeneous nature of the information sources, is mainly caused by the ongoing demand in the field of information presentation. The new powerful platforms that have emerged with the new data models such as the object-orientation which supports voice and video has changed the view of the new information requirements. The situation ends up with new powerful information platforms coexisting with old legacy systems using hierarchical, network, and relational data repositories that cannot be discarded for many reasons. Most of which relate to the huge amount of information stored on them and the cost of translating and moving the information to new platforms. Sometime, even if the data could be migrated to new platforms, the rewriting of the code may become almost an impossible process, which could be too costly. Furthermore, the information representation may differ in the similar function systems, which may be due to the differences in thinking between human beings. Sometimes, the representation of the information is derived by many factors such as the data owners` business requirements and other issues related to the data owner integration process of the new systems with the existing old systems.

Metadata, of course, is the data about data. In software design, the metadata are the schema of a database such as the tables, relationships between tables, the types and characteristics of the data the tables contain including the different constraints govern the data. In object orientation this is equal to the class diagram of an object-oriented system such as classes, relationships, inheritance, attributes and methods. In political design, the metadata are the Constitution under which the system operates, any amendments to it, any laws clarifying or implementing the system, judicial decisions explaining it, and (least important of all in a government "of laws, not of men") policy statements and orders of the executive.

The current Internet web browsers are not designed to browse and support the interoperation of the heterogeneous schemas metadata information willing to interoperate. The cooperation means the heterogeneous information sources accessible to all the information consumers give the maximum freedom to them as if they are accessing information source of the type they are familiarize with. By such capability, information consumers can expand their own information sources by adding to them records from other distributed information sources having similar information, which will expand their information domain and will give better answers especially in the statistical health related information.

In this paper I am presenting a technical perspective on the system architecture requirements for the heterogeneous databases interoperability in the light of the current Internet advances. I am negotiating the different issues related to the metadata hosting that forms the bases for integrating and aggregating the distributed heterogeneous information sources. In a further stage I am also negotiating the linkage between the information sources and the permitted user groups. The major contribution of the work in this area is to create a unified methodology to advertise about any schema type that is accessible as an information space using the web browser as our communicator between the global information producers and consumers. The proposed solution will require a specialized proxy server responsible to communicate the cooperating information producers and also a specialized search engine responsible to keep the different information

0-7695-0680-1/00 $10.00 © 2000 IEEE

consumers informed about the available information space.

1. Introduction

Interoperability between heterogeneous information sources can be defined as the ability to generate a single virtual view of heterogeneous schemas without sacrificing autonomy. An important aspect of this is to coordinate changes of such autonomous schemas while providing guarantees on quality of the different schemas information flow and the quality of the different services on the schema.

Although obvious, it is worthwhile to restate that we will be using the sentences data source, information source and schema interchangeably to mean a database. Also, interoperability phrase may be used for the hardware and for the software. For the hardware it always means the capability of connecting different vendors hardware. In software it has many faces. Two of the most important are application and database interoperability. Reuse of the legacy applications and their attached information sources are the most frequently quoted requirements for application and data interoperability. The largest problems still challenging the area of software interoperation are the syntactic and semantic data interoperability.

Of course, interoperability between the heterogeneous and distributed information schemas still suffering from partial of the above mentioned problems which has been limited by the new advances happen in the area. In that, the new advances has shown that through the use of the proper mediators most of the interoperation problems could be overcome. Also, the new advances in encapsulating parts of the legacy software have also an added value in the merging advances of the legacy and new database applications.

The heterogeneous database interoperability means openness of any type schema to any type schema for all the database operations. We do mean by the database operations add, delete and edit. Assume, if an information source user is using a hierarchical database, he can do any database operation on any type remote database. In this case the word cooperation may be closer to the meaning we are working towards to achieve, which is wider than the normal interoperability. The most basic shape of cooperation is to have an information source, which cooperate with other heterogeneous information sources, provide similar information to form a single unified information source similar to the requester schema. In cooperation both the information producer and the information consumer should apply certain standard to achieve the interoperability, which is the discussion highlighted in this paper.

We think the overall work in the area of database interoperability is nearly directed towards achieving number of strategies that are necessary for accomplishing the interoperability process among the distributed heterogeneous databases. The initial main strategies that have been considered during our work in this project are:

1. Initiate the strategy of metadata registration in the cooperating information sources that acts as the infrastructure, which will be used by both the information produces and the information consumer to advertise about the existence of certain shareable information.
2. Binding the local schemas with the global cooperating schemas and allow incremental addition to the cooperative module without any changes in the local queries using the IE knowledge.
3. The requirements towards providing database services in the current Internet browsers using specialized metadata management capable proxy servers. The relationship between the different components attached to the proxy is shown at figure 1. In that, web information management is one of the most important current trends in data management technology and research area [6].
4. Provide the necessary management services for the interoperation such as security services; syntactic/semantic data dictionary services; history tracking and the replication services.
5. The initiation of a universal object identification strategy to provide the maximum flexibility to the interoperation objects that will be used from anywhere.
6. Design the infrastructure of the security management, which acts as a unified security manager serving the local heterogeneous information sources.

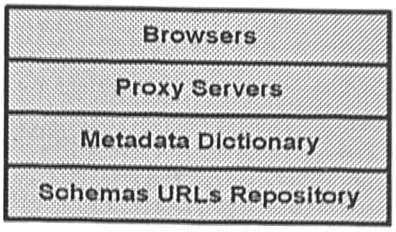


Figure 1

It is also obvious that the real major database interoperation problem is related to the information required by any one to be able to join the cooperating data sources, in a way to make the interoperation process done as transparent as possible. No one person can be considered familiar with all the different data sources and that he may access only the information relevant to him.

0-7695-0680-1/00 $10.00 © 2000 IEEE

Such process considers that person has the required knowledge about the information space of his interest, which may exists overall the globe. Also, he should have a way by which he is informed about any changes so that he stays up to date and gets the required services from the remote information producers. Because of such difficulties, the problem related to the mechanism of dealing with the rapidly increasing sources of data is significant. In order to cope with the rapid increase data sources environment there should be a solid mechanism based on a unified policy by which both the information producers and consumers follow to update each other. In that, among the huge amount of data available over the global network a very small amount is of interest to interoperate with other data sources. As long as this problem is not being dealt with, its breadth will become uncontrollable. I think, as a first step, having the required knowledge about the cooperating databases is the major solution to the problem.

As part of the proposed prototype a web-enabled database interoperation proxy server can be used to form a virtual community, where participants in remote locations can exchange metadata information about the cooperating information sources in electronic format. The existence of such facility will have many advantages by which the most important is to have a communication point between the information producers and information consumers. Figure 2 shows the initial metadata browser components we are looking forward to implement based on the initial design appear on appendix A.

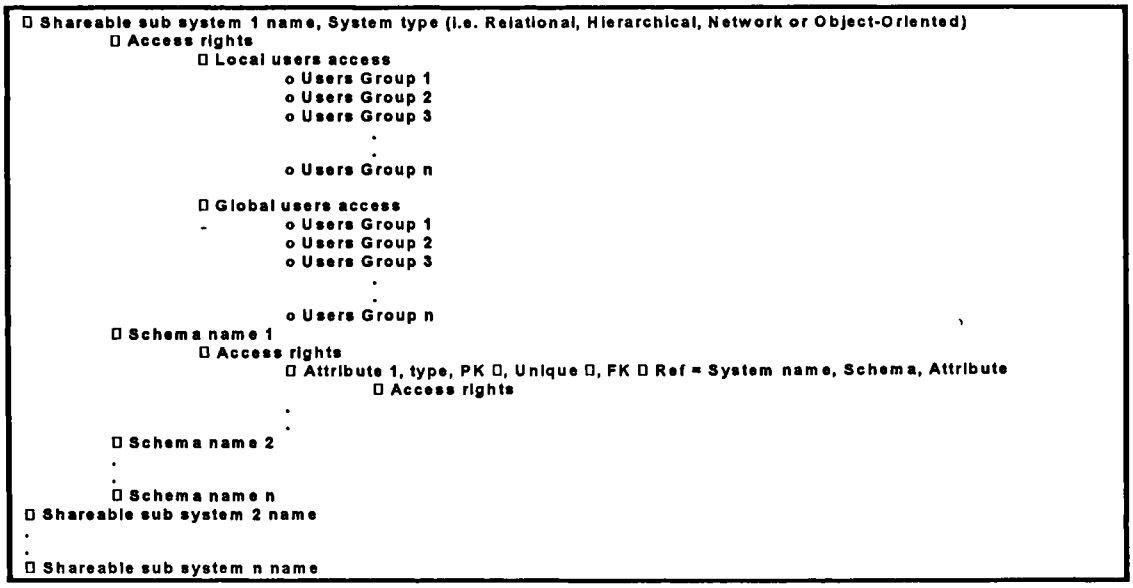


Figure 2

The overall goal of the work in the database interoperation is to demonstrate a new, modular and dynamic framework for the knowledge handling from the information producers and consumers side in the distributed, autonomous and heterogeneous databases that is necessary for the interoperation process. The framework will consist of the necessary parameters that should be known to the interoperating information sources and how the local database applications will be able to interoperate with other heterogeneous information sources via a proposed layer. A key aspect of making the knowledge about the interoperating databases available to the information consumers involves assigning a unified metadata handling protocol, which is imperative for developing methods that can perform tasks with as little human intervention as possible. Such interconnected information sources supporting the sharing of data may be called cooperating information sources by which the design, construction, use and evolution of such system within the above paradigm will require sophisticated technologies from many different areas of computer science.

The cooperation between the heterogeneous distributed data sources can take two distinct paths. The first, which forms the major demand from the interoperation of databases, belongs to the normal information gathering needs. This means the requesting user is having an information pool by which he needs to share it and cooperate with other people owning similar information so that he can get better statistics for better business decisions. Such type of requirement increases in areas like the medical statistics. The second path is considered when an information source allows updates from the global either by anybody or by specific people. This type of operation is very much similar in nature to some existing Internet HTML services. In reality it is an advertisement about existence of information pool and cooperation between information sources of possibly different schema types. The difference in the normal HTML services is they don't know about the other information sources available around. While in the case of the IE the information source will be open in a sense to give the information services to the information consumers in a way by merging their information pools with other information owner pools

having similar information. In the case of the IE the distributed information consumers will be informed about the available information they can access from the global information producers sources. The information consumers will be able to use their own data entry screens of their database applications rather than having to use a pre-prepared data entry screen which means they will not be able to merge their information pools with the other related information pools. Furthermore, when adding new record or editing an existing record is allowed than this type of operation should apply the stipulated constrains assigned by the information owner.

2. Related wok

The need for software to support data distribution has resulted in the development of the distributed database management software, which has itself been given impetus, by the rapid developments in telecommunications and network technologies. Distributed databases are a collection of multiple, logically interrelated databases that are distributed over a computer network, together with a distributed database management system which is a software system through which distributed databases are managed and through which the distribution is transparent to the user. Some distributed databases are homogeneous in nature, which means that the local database managers composed of a single DBMS product. Others are heterogeneous in nature, where different DBMS products make up the local data manager group [14].

The most familiar names describing differences in Database Management Systems data models are "heterogeneous databases", "distributed databases", "multidatabase systems", and "federated databases". They relate to the problem of making different types of databases communicate and exchange data between each other in such a way keeping all performance, security, reliability, and simplicity to an acceptable level for the users. In that, systems will mainly provide a standard interface, which will make it possible for the databases to interoperate with each other. Systems such as Sybase provide an open interface, which includes commands such as "begin-transaction", "prepare-to-commit", "commit", and "abort". Similarly, Oracle provides an "explain" command which can be used to ask the system how it has optimized a given query. The information obtained this way can be used in global query optimization. Also, Ingres has released the open Ingres version which is a distributed Information manager that allows a user to treat information resources within an organization as one unified information resource with the characteristics of a single local relational database. The data, which is accessed by Open Ingres in any local or remote databases, should be completely supported by Ingres. None of those is considered as the complete solution to the database interoperability problem. Early work on these topics indicates that they need to provide integration mechanisms for schema and process, while most of the work to date has concentrated only on schema integration.

Up to the best of our knowledge works such as FINDIT [1] and DIOM, stands for Distributed Interoperable Object Model [5], has addressed the problem of finding information in a large scale network of autonomous and heterogeneous databases and educating users about the available space of information. FINDIT mainly intended to advertise locally about the shareable local data sources. In this case, global users need first to send queries to remote information owners, and second the local user query will be sent in an indiscriminate manner to unknown database server. This is because not every database server knew about the available global information space it can access. The only available shareable information to the local site is the database servers linked directly with the local user server of that site. User servers are linked together so that they can exchange information. The local user query in this case may traverse to the outside world and may get rejected or the result of the query may not form any interest to the sender. By this it cost the user the time between the query is sent and the answer is received.

Recently, substantial attention in the DBMS area has been attracted to the problem of heterogeneous database integration systems design. The practical importance of this direction consists at the development of tools making possible the coexistence of different DBMS supporting various data models, which meet applications requirements. The methodological importance of this direction arises from the fact that integration of heterogeneous databases requires methods of equivalent data model transformation and methods of constructing unifying data models and languages promoting generalization of various approaches to the development of DBMSs programming language [7]. Because of this, a number of projects have been established in this area such as the Jupiter System, which is a prototype for multidatabase interoperability. In this system, the majority of the work has been done on the mapping mechanism between the different schema of the databases. The mapping of schema have been considered between autonomous and possibly some heterogeneous databases [11]. Also there are other recognized projects in this area such as IRO-DB and ESPIRIT III. They providing a method of integrating heterogeneous data sources from the design perspective and the data perspective. They allow for the integration of heterogeneous object-oriented and relational DBMSs [1].

There are some other approaches written especially for newly developed object-oriented database interoperability and as an extension they also capable of importing and exporting data from and to other databases [2]. The *Common Object Request Broker Architecture* CORBA standard has been designed for the newly designed applications, and is considered as a standard to let objects over a heterogeneous environment to talk to each other

regardless of the communication protocol used in each side of the internetwork [4]. As indicated in the literature, the real problem concerns legacy systems. CORBA is considered as a significant step in satisfying the needs of applications, and it makes a substantial contribution to interoperability and provides a high-level view for developing and integrating applications. However, it is limited to the classical object model and lacks an explicit view of resources [8]. The real significance of CORBA specification is for application developers who want to build new client/server applications that will work across disparate platforms.

The Meta Data Coalition Open Information Model MDC OIM version 1.1 proposal [10] is also a close definition to the metadata registration we are looking for. At this stage we can see that our proposal is forming an extension to the MDC OIM. This extension is considering the metadata and data exchange mechanism and forming handling policies between both the information producers and consumers. We also, design an interoperation interface that is using the metadata to link the different heterogeneous information sources together. Through the use of objects the cooperating information sources should dedicate objects responsible to transfer the records between the information sources in case if one of the normal database operation is to be fired to the information source.

Today Wrappers programs enables character or console based programs, that have difficulty running in a telnet or terminal session, to run in another platforms. Even though wrapping every scalar code would result in much lower performance than using the primitive types, but it is simple to implement.

The main objective of the database interoperability is to merge different schema type information sources together, which may requires consider different parts of the information source for the various types of users accessing it. Although wrapping techniques will simplify the mission for the total schemas usage based solutions where all the database fields are used as if the accessing users are working at the same machine the original databases are loaded on them. They are unable to handle the various accessing users' requirements unless either creating multiple functions to meet the different requirements and wrap each of them individually by different name. Or on the other hand, define another layer having access to the different users requirements and the data sources breakdowns which is exactly the Interoperation Engine proposed layer.

In addition, much of the work in the interoperation of the distributed heterogeneous databases [15, 9] is also relevant. However, our viewpoint about the autonomy of the distributed heterogeneous databases and how it is suppose to be handled is different from the previous methods. Instead of applying static interoperation, which may effect the local autonomy, we give the full autonomy to the information owner to decide about what parts of his data to be in the interoperation and who should use this data. We utilize the advances in the Internet and propose a prototype act as the basement for the database metadata management in a way similar to the existing HTML files management.

3. Overview of the IE

The most important is that the approach of databases cooperation ensures no effect on the other normal services offered by the database management systems. Among those services are: schema translation management, programming language translation management, semantic inconsistency management, and other aspects related to the operating system and the communication protocol layers which are considered as outside the database research realm.

All these issues make a strong demand towards having a new dynamic mechanism for the cooperation of the heterogeneous distributed data sources. In that, this issue needs to be looked at in a new way. Although there are currently a number of suggested static definitions for the cooperating data sources, no one is considered as the ideal solution for the rapidly increasing number of cooperating heterogeneous distributed data sources. Disco project [15] is one example of the static definition of the necessary databases interoperation parameters and which our view is considered as a dynamic implementation of parts of the Disco.

We feel that at this stage it is important to provide an infrastructure capable of linking the heterogeneous distributed data sources in an incremental manner and for only those related portions of the data sources. This infrastructure is also responsible for sending queries to all the cooperating sources asking for certain information of interest. Also, this infrastructure should deal with all the users in the globe having access to its local data sources. Our main contribution by this work is an infrastructure for an interoperation atmosphere serving the heterogeneous distributed databases and having a behavior similar to the Open Shortest Path First OSPF [12] routing mechanism.

OSPF is a routing protocol used within larger autonomous networks in preference to the Routing Information Protocol RIP, an older routing protocol that is installed in many of today's corporate networks. Like RIP OSPF is designated by the Internet Engineering Task Force IETF as one of several Interior Gateway Protocols IGPs.

Using OSPF, a host that obtains a change to a routing table or detects a change in the network immediately multicasts information to all other hosts in the network so that all will have the same routing table information. Unlike the RIP in which the entire routing table is sent, the host using OSPF sends only the part that has changed. With RIP, the routing table is sent to a neighbor host every 30 seconds. OSPF

multicasts the updated information only when a change has taken place.

We propose a middle engine, which can act as an added-on facility on the Internet browsers, responsible for binding only these heterogeneous distributed data sources of interest. This middle engine will not require the set of related applications to agree on one global view. Our approach is based on the information availability and information demand. It will also depend on the information advertisement technique for the purpose of advertising an available information source. The infrastructure will also form the foundation for all the supporting areas in the distributed databases to take place towards the supporting of heterogeneity. The following Figure 3 is explaining the proposed steps to be undertaken for registering the metadata, advertising about the shareable parts of the local information sources and delivering the information parts to the distributed information consumers.

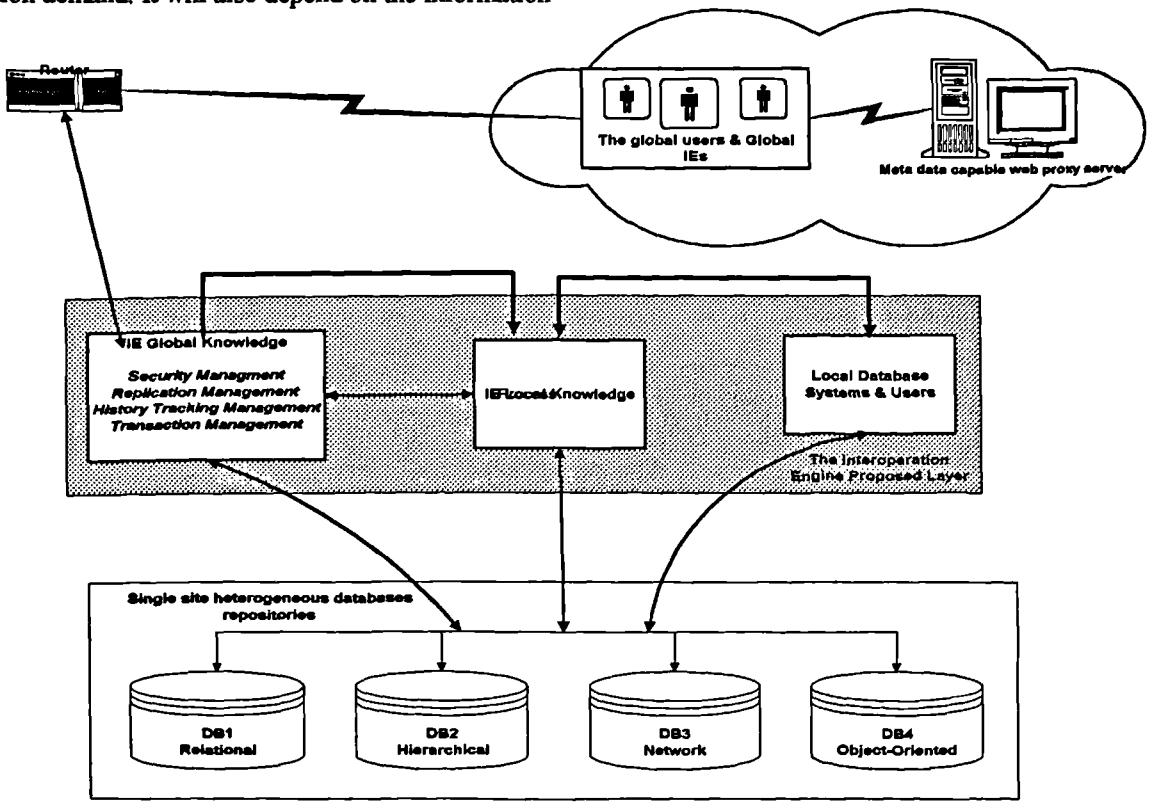


Figure 3

Further, figure 4 presents the connectivity between the underlying system components and the functional relationship in the IE in a detailed and concrete fashion. The user can have his own domain where he can create his own data sources access profiles by which he can access them in his own applications. Also, as another option he can rely on the profiles created by his local DBA. In that the DBA in any site is the link between the local information consumers and the global information producers.

3.1 Querying for information

Both local users and the local DBA for gaining certain information can submit queries to the local IE. The local data dictionary is the main part where all the submitted local queries are processed. When a query is submitted, the user is given the available information space that answers the query and the existing data sources access profiles by which he can use in his application or query code. There are some other cases by which some information owners may advertise about the availability of certain information in their sites without giving access to any global information consumer until a request is received asking for such access. Information owners may give access to global information consumers based on, for example, the further mutual benefits they may gain or for any other security purposes.

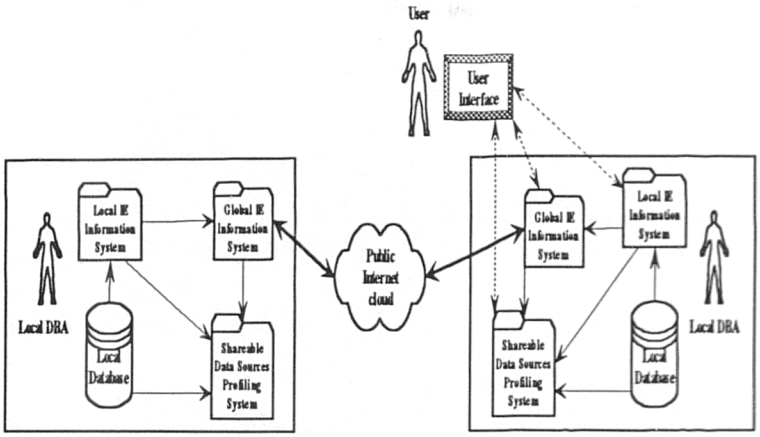


Figure 4

3.2 Exchanging information between participating IEs

The local IE information system contains a definition of a complete local systems. A further definition for the shareable parts of the local systems are defined as routes and only these parts are transmitted to the global IE information system. In each of the cooperating sites the global IE information system link the local shareable subsystems with the other global cooperating databases. Any alteration in the global shareable space is mainly done locally in the local IE information system and then migrated to the global IE information system. This stipulation is for the purpose to make the global IE information system maximum reliable and to make sure no long intervention is to occurs in the cooperation responsible subsystems. The initial partial local IE information system UML based classes are shown at appendix A figure A.1 [3, 13]. Accordingly, figure A.2 shows the global IE information system.

The shareable data sources profiling system is where database profiles are created and accessed by the local application programs. Basically it links information about data sources from both the local and global IE information systems.

Although, the design of the IE in its preliminary stages, this will involve assigning an information handling strategy between the cooperating data sources by which the exchange of the updates for the necessary metadata between the information sources can be done. The strategy should guaranty the maximum reliability and minimum intervention to the interoperation services.

4. Description of Participating Components in the IE

The actual building infrastructure of the IE is assumed to be the Internet by which nobody owns the backbone. Figure 3 has shown the overall architecture of the different IE components and how they are liked together. As shown in the figure, the main components are the Interoperation Engine layer, the heterogeneous databases repositories and the metadata proxy server.

Subsequently the *IE* local knowledge consist of three interrelated components of which each plays an important rule in the success of the interoperation mission of the distributed heterogeneous databases. The three interrelated components are (1) u*sers and shared systems profiling*, (2) *heterogeneous schema management*, and (3) *cooperating schemas profiling*. Those are explained graphically at appendix A figure A.1.

4.1 User and shared systems profiling

The IE system supports two type of users. These are the ordinary information consumer and the IE administrator. The one who will mostly benefit from the IE is the ordinary information consumer. The administrator is the person who will create users profiles, systems profiles and access profiles. User profiles are the grouping of the local and global users according to local site policies. Systems profiles are the same as user profiles but on the local and global systems. Access profiles are the link between a user profile and systems profiles. The following example explains the profiling technique in the IE system including user management. Figure 5 shows the relationship between the class diagrams for both the local and global profiling management subsystems. Details of the participating components shown at appendix A figures.

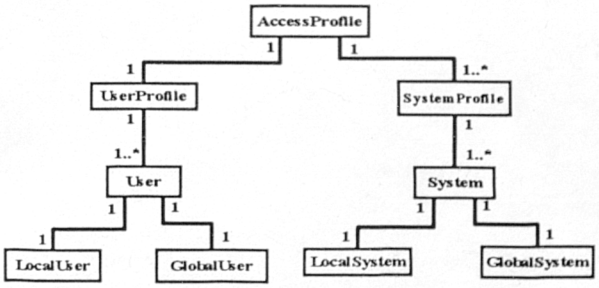
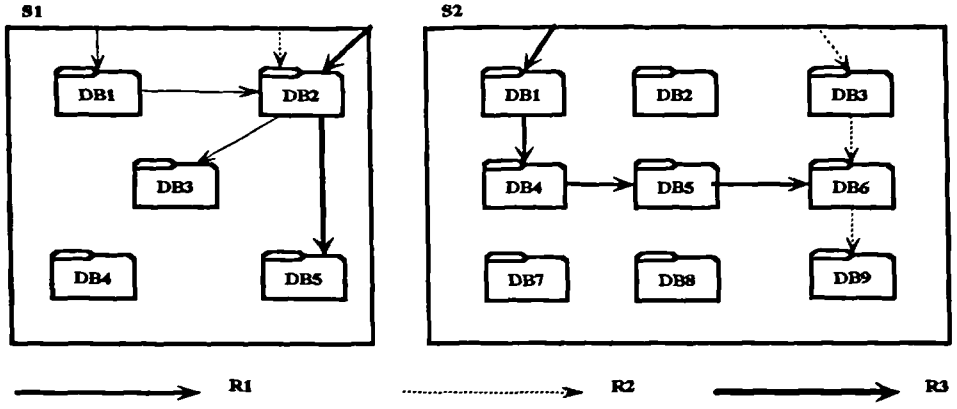


Figure 5

Initially users and user profiles are defined locally in the local profiling management subsystem for all users of a single site. Only those whom will share global data sources are copied in the global profiling management subsystem. The local administrator in each site is able to access the global profiling management subsystem to assign global users to local system profiles. Also, he is the one who creates the cooperating schemas profiles to be used by the database application programmers, which is thoroughly discussed further in section 4.3. In the above breakdown shown at figure 4 the IE is intending to send only the shareable parts of the information sources to the global information consumers. Access permissions of the global users on the local information sources will be checked two times. The first in the query originator IE and the second in the target IE and the synchronization and accuracy of the applied information will be checked prior to the query submission. Queries or requests are only passes if and only if information in both sites the same.

Having dedicated information for the local site and other for global sites will increase security and reliability of information. In that, local site will have a total information about the existing information systems. However, only the part of the information sources suppose to be shared with the global users are copied in the global profiling management subsystem.

As indicated earlier, profiling simply means grouping. Profiling is one of the proposed facilities to be provided by the IE. The main purpose of profiling is to make the work more controllable, manageable and traceable. The IE is mainly supporting three types of profiling techniques: user profiles, cooperating system profiles, and access profiles. User profiles are simply grouping users into groups according to different policies such as they have similar access on certain systems or maybe according to user's original site. Cooperating systems profiles are the parts users will get access on them. In that, each site may have many systems by which each system has number of subsystems forming the possible accesses in a local database system. Each group is known as a route in the IE. The following figure 6 explains how single site schemas will looks like.



Local IE systems and the routes supported by each system

Figure 6

As shown in figure 6 the local site is mainly supporting two systems (will be called as S_n for system and the system number) where each is having number of routes to be accessed by the global users R_n. Further, DB1 may consists of number of tables (t_1, t_2, t_3, ..., t_n). Tables also will consist of number of attributes (a_1, a_2, a_3, ..., a_n). System 1 as shown supports three routes.

S_1: (R_1, R_2, R_3)
S_2: (R_1, R_2).

The following is the definition of the routes in each of the two systems.

$S_1.R_1$: (DB1, DB2, DB3)
$S_1.R_2$: (DB2)
$S_1.R_3$: (DB2, DB5)

$S_2.R_1$: (DB1, DB4, DB5, DB6)
$S_2.R_2$: (DB3, DB6, DB9).

On the other hand the local site have got four users profiles as the following.

UP_1: (U_1, U_2, U_3, U_4)
UP_2: (U_1, U3, U_7, U_9)
UP_3: (U_{10}, U_{11}, U_{12})
UP_4: (U_{50}, U_{70}, U_{90})

Also, the local site has decided to establish the following system profiles (SPs) on the two defined systems from the defined routes according to private information policies.

$S_1.SP_1$: ($S_1.R_1$, $S_1.R_2$)

$S_1.SP_2$: ($S_1.R_1$, $S_1.R_3$)
$S_1.SP_3$: ($S_1.R_2$, $S_1.R_3$)

$S_2.SP_1$: ($S_2.R_1$)
$S_2.SP_2$: ($S_2.R_1$, $S_2.R_2$)

Once system profiles and user profiles are defined this will make connecting users with systems an easy task. The following is the connection of user profiles to system profiles, which is called access profile AP_n.

AP_1: UP_1 can access $S_1.SP_1$, $S_1.SP_2$
AP_2: UP_2 can access $S_1.SP_1$, $S_2.SP_2$

Here, any additional system profile that will be added later to the user profile will be added to the list related to that user profile.

Until now the consideration is that users are permitted access on the full schemas defined as DB_N. In some cases the database owner may need to specify the grants a step forward by giving access to certain attributes within some schemas. In this case he should specify routes contains certain schemas and attributes. These will be defined in the global shareable subsystem in each local IE and fired to the global IE side when required. Our past example of figure 6, S_1 is mainly supporting three different routs, which appears in the global IE as different systems entities. If we recall $S_1.R_1$: (DB1, DB2, DB3) and assume that DB1 has three attributes (A_1, A_2 and A_3), DB2 has four attributes (A_1, A_2, A_3 and A_4), and DB3 has again four attributes. Here we still can define more than one route on the same three databases showing each time different attributes to be supported by each route. In this case the subsequent details which will appear in the route definition are the attributes supported by each route. The following is the definition of the routes in each of the two routes of S_1.

$S_1.R_1$: (DB1[A_3], DB2[A_3, A_4], DB3[A_3, A_4])
$S_1.R_4$: (DB1[A_2], DB2[A_2,A_3, A_4], DB3[A_2, A_4])

4.2 Heterogeneous schema management

As indicated earlier, the local administrator is the one who make the data sources ready to application programmers by creating the required application profiles and attach them to the user profiles as access profiles. In this case application programmers access system profiles in their applications rather than directly accessing the original data source. This step has got many facilities of which the application programmers does not need to change in the application source code when new data sources added into the cooperation process. The following few paragraphs are explaining in an example how the distributed heterogeneous information sources are managed. Also, explain how application programmers should deal with the profiles of the IE in their applications.

The cooperating data sources are either contains similar or dissimilar information. If they contain similar information then the possibility operations between them is unification, intersection or difference between the records. In the case if they dissimilar the only possible operation between them is relationship through some common field in both that can be considered as a primary and foreign keys.

As indicated earlier, the cooperating information sources will be managed through the global schema profiling subsystems. It will be responsible to apply the three operations in between the master information sources with the other distributed information sources. The three operations are unification, intersection and difference. Figure 7 shows preliminary contents of the *global schema profile.*

Profile name: ABC system schemas profile

Local schema name: DB1
Operation: Union
Involved schemas: S1, S3, S4, S7, S8, S9, S20, S34, S99

Local schema name: DB2
Operation: Relationship
Involved schemas: S199, S333, S114, S237, S548

Local schema name: DB3
Operation: Intersection
Involved schemas: S101, S301, S401, S701, S801, S902

Figure 7

As figure 7 shows, if the case is relationship as the operation applied on DB2, there will be two different possibilities. The first, if the relationship is to be applied with a single remote schema than this is considered as a normal case. The second, which is the abnormal, if the relationship is with multiple remote schemas than this will require apply an operation on the cooperating information sources to get them as a single information source prior to setting up the relationship linkage.

The proposal is the application program not directly reading from the remote data sources. Rather, it is done through the *global schemas profile* facilities provided by the local IE, which is mainly liaising with all the related global schemas in a manageable, expandable, adjustable and dynamic manner. From the operating systems side there will be some extended requirements to make the cooperating information sources linked together, as well as, synchronize with each other. This task is purely related to the operating systems sense they will be responsible to share memory and open files between the operating system processes, protect pages of memory that has the cooperating information sources records and linking the distributed cooperating information sources memory pages together considering the sorting constraints.

The main advantage of this approach is the creation of the cooperation atmosphere between the different information sources is done in an incremental manner. Additionally, this cooperation process will not involve changing in the already compiled and tested source. In that, the user will

only change in the profile related to the system using the IE provided facilities.

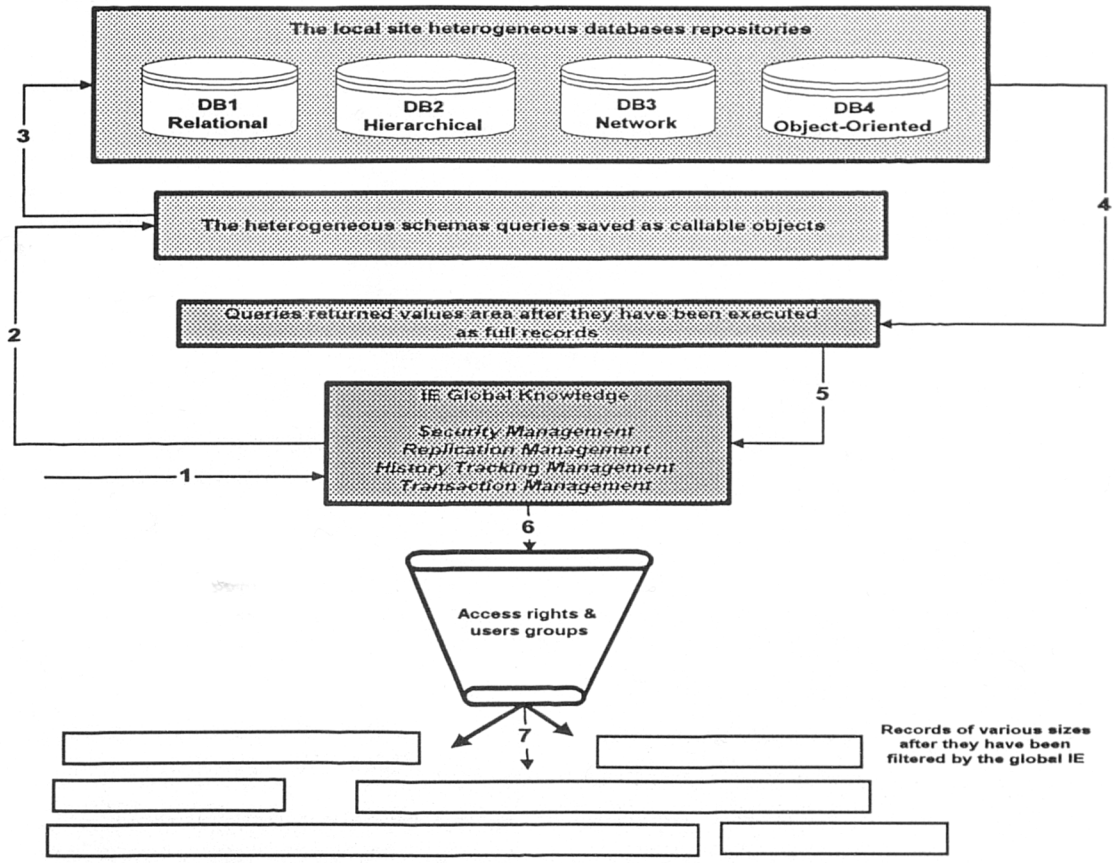


Figure 8

According to the IE stated design, the information producers will not be able to interrupt any running query against their information sources. The changes in the permitted information space will be done into two stages. The first stage will be the changes on the permitted information space prepared through the local IE. Once the changes finish it will be migrated to the global IE side and the old information space will be replaced with the new one. The following figure 8 clarifies how the IE deals with the queries and where the filtration against the access rights are applied.

This technique also has got another advantage by which the global part of the system will not be interrupted, as well as, modifications in the available information source for the global can be accomplished in the local site and then fired to the global. Changes will take place and the necessary parameters in the global IE services will be modified.

4.3 Cooperating Schemas Profiling

As explained earlier, application programmers depend on accessing data sources profiles rather than actual databases. By means these data sources has to be ready to them so that they can call them from within their application. Databases are called as one at a time from within any DBMS programming language. The database administrator is the one who normally builds the data sources profiles that may consist of more than two information sources. He is the one responsible to give the information consumers the different reports about the provided information source they gain from the global information owners. The plan is also to let the individual users having access on global information sources to create there own cooperating schemas profiles.

In this stage the data sources profiles are made ready to be called from within the application program. Assume that each of the three databases has number of attributes, and the first database DB1 that our example is about has three attributes (A1, A2 and A3). As a normal case the three attributes may be accessed by their names in the application program. In the case if there are other cooperating databases with DB1 than the access to the attributes is through the use of the data sources profile. Assume our local system is able to cooperate with other three global systems having equivalent data as our local

system. Those are S2, S3 and S4 respectively. Assume again S2 database 1 has four attributes (A1, A2, A3 and A4), S3 database 1 has again four attributes (A1, A2, A3 and A4) and S4 database 1 has five attributes (A1, A2, A3, A4 and A5). The definition step of the equivalent global attributes to our local attribute as the following.

Local $A_1 = S_2.DB_1.A_1, S_3.DB_1.A_2, S_4.DB_1.A_4$
Local $A_2 = S_2.DB_1.A_2, S_3.DB_1.A_4, S_4.DB_1.A_1$
Local $A_3 = S_2.DB_1.A_4, S_3.DB_1.A_3, S_4.DB_1.A_3$

There are tow possibilities when liking the cooperating data sources with the local application. The first is when local application accesses a local database as in the above example where local database cooperates with global systems S2, S3 and S4. The second if the local application doesn't have any local database and will directly access remote global databases where in this case the local attributes will be considered as the local application parameters.

5. Metadata Handling Protocol in the Distributed IEs

The Extensible Markup Language XML [16, 17] will have much input to the interoperation process. In that, the XML has added a powerful transport mechanism to the rapid database interoperation requirements. Sense the different information handling mechanism between the different interoperating information sources can use this language as a method for putting structured data on a text stream. The transmission of information should depend on the policies that are assigned by the proposed interoperation engine IE when applying the different database operations.

Handling protocol of exchanging the metadata within the cooperating database systems is considered as a strategy task by which all the cooperating sites have to agree on certain handling methodology. This section does not form any comparison to any international standard rather than considering a methodology that fits the different requirements of the heterogeneous database interoperability. We understand that the functional programming languages may contribute in the design of such protocol, but at the time being this is beyond the scope of this paper. For the sake of this paper, we are proposing a protocol that we see it will fit the initial requirements defined by the IE prototype. The protocol, which is XML based [16, 17], is assumed to deal with the shareable database systems metadata and all the related information for the purpose of exchanging this information for all the cooperating IE sites. Sense every schema type has got its own definition procedure with different constraint assignments it would be difficult to provide a unified protocol to handle all the schemas. As a first attempt we are planning to tackle the general protocol shape in this section. As a further step we are planning to extend this protocol to make it as unified as possible to handle any schema type requirements. The following figure 9 explains the basic required information to be processed by the proposed protocol.

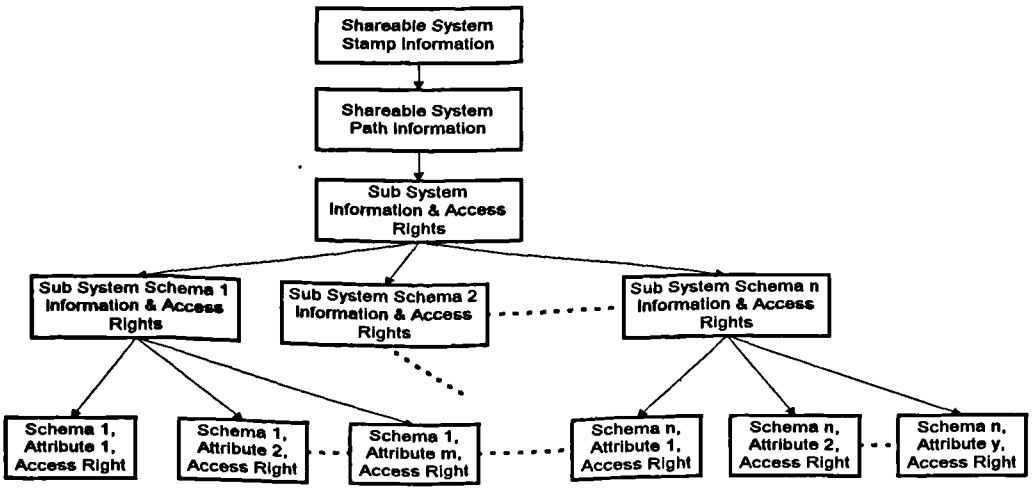


Figure 9

Based on the above figure, if we consider the relational schema shareable system defined at figure A.2, the global IE, than we will need to define different components to cover the whole schema design.

6. Conclusion

In this paper, we presented an architecture by which a distributed and heterogeneous information sources can cooperate together in an interoperation atmosphere to share information taking into consideration the current advances in the Internet facilities. More specifically, we describe a mechanism that enables information producers and consumers to have a common pool by which they know each other and the information space provided for each.

This mechanism accomplishes the goals without any effect on the autonomy of the participating information sources. The plan also is to conduct more research in refining this architecture. We have implemented some parts of the prototype and working to implement the prototype as a whole to validate the design ideas presented in this paper.

In this paper, we address several issues for sharing information in a cooperative manner across autonomous distributed heterogeneous information sources. Historically known that the more there is sharing, the less autonomous databases are. For instance, the use of schema integration increases data sharing dramatically while bringing database autonomy to nonexistence. In contrast the above-described prototype designed to increase cooperation between the distributed heterogeneous databases without dimensioning database autonomy. Our approach provides a mechanism that enables database application users to be informed about the available information space they can gain from the globe information producers. It also enables them to share information with other information holders in a transparent, expandable and autonomous manner. The design of our prototype is done in a way to contribute with a metadata capable web proxy servers so that they become heterogeneous databases accessible web browsers.

References

[1] Ralph Busse, Peter Fankhauser, Gerald Huck, Wolfgangklas, "IRO-DB : An object-oriented approach towards federated and interoperable DBMS", Proceedings of the International Workshop on Advances in Databases and Information Systems ADBIS'94, Moscow, Russia, May 1994.

[2] Ralph Busse, Peter Fankhauser, Erich J. Newhold, "Federated Schemata in ODMG", Proceeding of the Second International East-West Database Workshop, September 1994, Klagefurt, Austria.

[3] Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide", Addison Wesley, 1999.

[4] K. Burleson, "Managing Distributed Databases", John Wiley & Sons Inc., 1994.

[5] A. Cardenas, M.H. Pirahesh, "Database communication in heterogeneous database management system network", Information systems, Vol. 5, No. 1, 1980, pp. 55-97.

[6] Asuman Dogac, M. Tamer Ozsu, Ozgur Ulusoy, "Current Trends in Data Management Technology", ISBN 1-878289-51-9, 1999.

[7] Leonid A. Kalinichenko, "Methods and tools for equivalent data model mapping construction", Advances in Database Technology-EDBT '90.

[8] Dinesh C. Kulkarni, Arindam Banerji, David L. Cohn, "Operating System Support for Cooperation in Distributed OODBs", "Distributed Object Management, by M. Tamer Ozsu, U. Dayal, P. Valduriez", Morgan Kufmann Publishers, 1994.

[9] Ling Liu, Calton Pu., " An Object-oriented Approach to Interoperable Heterogeneous Information Sources", Invited Paper in Proceedings of the Seventh International Hong Kong Computer Society Database Workshop, Hong Kong (May 1996) (Springer Verlag) pp49-65.

[10] Meta Data Coalition Open Information Model Version 1.1 Proposal, URL: http://www.mdcinfo.com/OIM/MDCOIM11.html, August 1999.

[11] John Murphy, Jane Grimson, "The Jupiter System: A prototype for Multi-database Interoperability", 12th British National Conference on Databases, BNCOD 12 Guildford, United Kingdom, July 1994 Proceedings, Springer-Verlag.

[12] OSPF and the Internet, URL: http://www.livingston.com/marketing/whitepapers/ ospf_whitepaper.html, Lucent Technology, 1999.

[13] James Rumbaugh, Ivar Jacobson, Grady Booch, "The Unified Modeling Language Reference Manual", Addison Wesley, 1999.

[14] Alan R. Simon, "Strategic database technology: Management for the year 2000", Murgan Kaufmann Publishers, San Francisco, California, 1995.

[15] Anthony Tomasic, Louiqa Raschid, Patrick Valduriez, "Scaling Heterogeneous Databases and Design of Disco", Proceedings of the International Conference on Distributed Computer Systems, 1996.

[16] Ronald Bourret, "XML and Databases", URL: http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases, December 1999.

[17] W3C Extensible Markup Language (XML) 1.0 Recommendation, REC-xml-19980210, URL: http://www.w3.org/TR/1998/REC-xml-19980210, 10 February 1998.

Appendix A

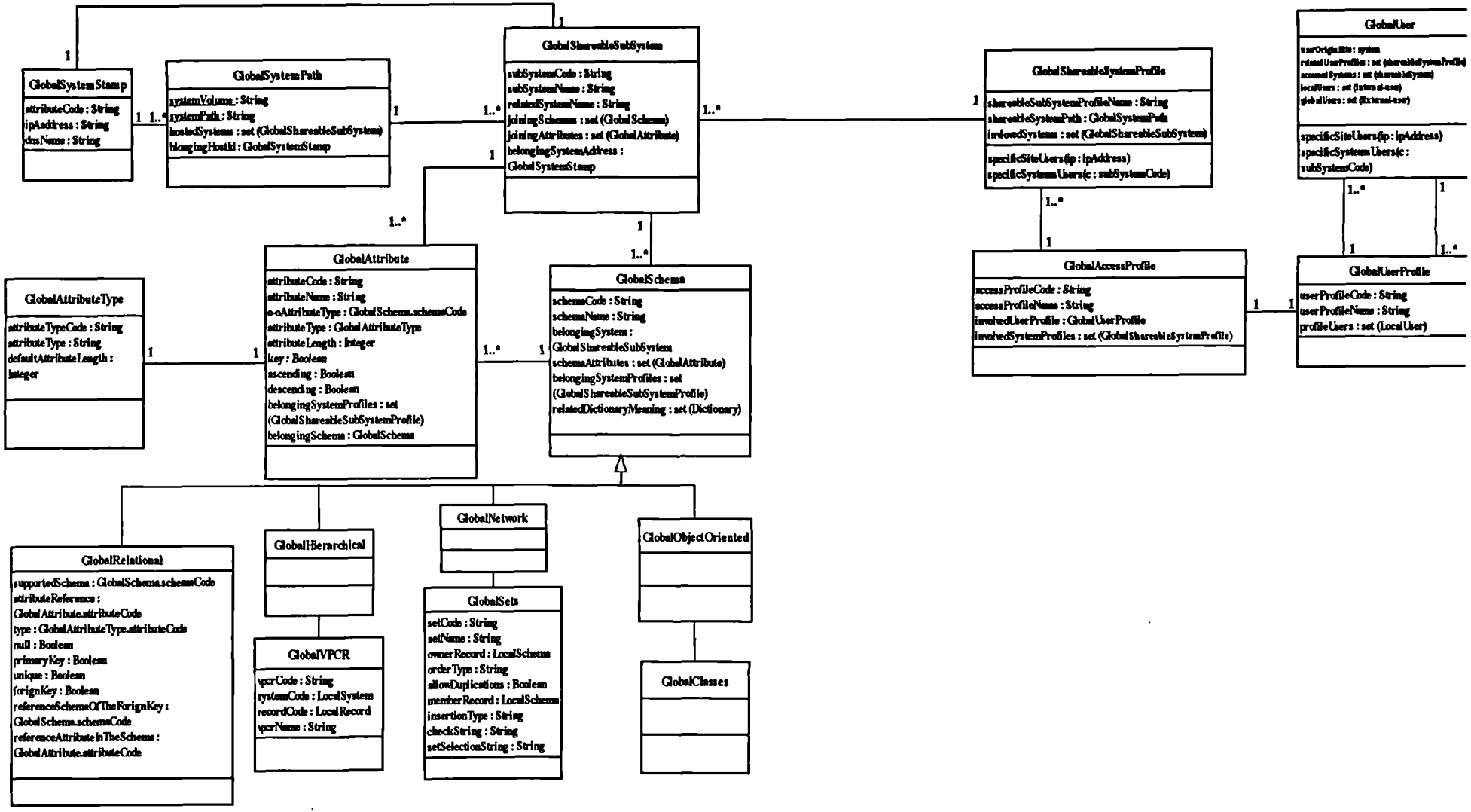


Figure A.1

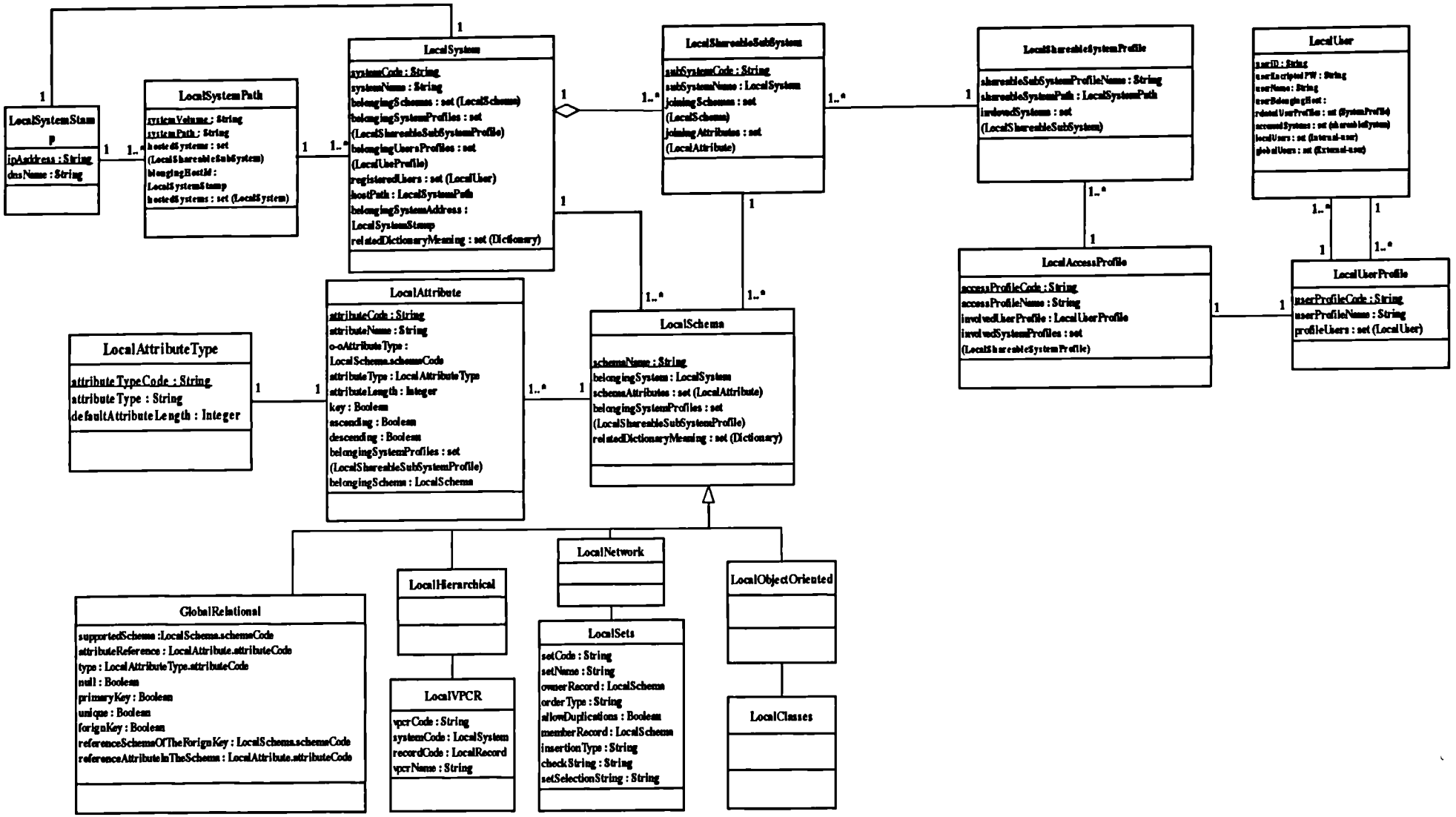

LocalSystemStamp
ipAddress : String
dnsName : String
LocalSystemPath
systemVolume : String
systemPath : String
hostedSystems : set
(LocalShareableSubSystem)
belongingHostId :
LocalSystemStamp
hostedSystems : set (LocalSystem)
LocalSystem
systemCode : String
systemName : String
belongingSchemas : set (LocalSchema)
belongingSystemProfiles : set
(LocalShareableSubSystemProfile)
belongingUsersProfiles : set
(LocalUserProfile)
registeredUsers : set (LocalUser)
hostPath : LocalSystemPath
belongingSystemAddress :
LocalSystemStamp
relatedDictionaryMeaning : set (Dictionary)
LocalShareableSubSystem
subSystemCode : String
subSystemName : LocalSystem
joiningSchemas : set
(LocalSchema)
joiningAttributes : set
(LocalAttribute)
LocalShareableSystemProfile
shareableSubSystemProfileName : String
shareableSystemPath : LocalSystemPath
involvedSystems : set
(LocalShareableSubSystem)
LocalUser
userID : String
userEncriptedPW : String
userName : String
userBelongingHost :
relatedUserProfiles : set (SystemProfile)
accessedSystems : set (shareableSystem)
localUsers : set (Internal-user)
globalUsers : set (External-user)
LocalAccessProfile
accessProfileCode : String
accessProfileName : String
involvedUserProfile : LocalUserProfile
involvedSystemProfiles : set
(LocalShareableSystemProfile)
LocalUserProfile
userProfileCode : String
userProfileName : String
profileUsers : set (LocalUser)
LocalAttribute
attributeCode : String
attributeName : String
o-oAttributeType :
LocalSchema.schemaCode
attributeType : LocalAttributeType
attributeLength : Integer
key : Boolean
ascending : Boolean
descending : Boolean
belongingSystemProfiles : set
(LocalShareableSubSystemProfile)
belongingSchema : LocalSchema
LocalAttributeType
attributeTypeCode : String
attributeType : String
defaultAttributeLength : Integer
LocalSchema
schemaName : String
belongingSystem : LocalSystem
schemaAttributes : set (LocalAttribute)
belongingSystemProfiles : set
(LocalShareableSubSystemProfile)
relatedDictionaryMeaning : set (Dictionary)
GlobalRelational
supportedSchema :LocalSchema.schemaCode
attributeReference : LocalAttribute.attributeCode
type : LocalAttributeType.attributeCode
null : Boolean
primaryKey : Boolean
unique : Boolean
foriegnKey : Boolean
referenceSchemaOfTheForignKey : LocalSchema.schemaCode
referenceAttributeInTheSchema : LocalAttribute.attributeCode
LocalHierarchical
LocalIVPCR
vpcrCode : String
systemCode : LocalSystem
recordCode : LocalRecord
vpcrName : String
LocalNetwork
LocalSets
setCode : String
setName : String
ownerRecord : LocalSchema
orderType : String
allowDuplications : Boolean
memberRecord : LocalSchema
insertionType : String
checkString : String
setSelectionString : String
LocalObjectOriented
LocalClasses

Figure A.2